THESIS

A STUDY OF THE EFFECTS OF SENSOR NOISE AND GUIDANCE LAWS ON SAM EFFECTIVENESS AGAINST CRUISE MISSILES

by

Murat Dogen

June 2015

Thesis Advisor:	Robert G. Hutchins
Second Reader:	Xiaoping Yun

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704–0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2015	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE A STUDY OF THE EFFECTS OF SENSOR NOISE AND GUIDANCE LAWS ON SAM EFFECTIVENESS AGAINST CRUISE MISSILES	**5. FUNDING NUMBERS**
6. AUTHOR(S) Murat DOGEN	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____N/A____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

Cruise missiles (CMs) are getting more advanced. To cope with the rapidly growing CM threat, national air-defense systems use surface-to-air missiles (SAMs) as interceptors. To intercept a CM with a SAM, an optimal guidance law should be used. Simulations that represent reality as closely as possible show the effectiveness of the missile system in various scenarios.

A three-degree-of-freedom, discrete-time, and three-dimensional simulation model that compares proportional navigation (PN), augmented proportional navigation (APN), and differential geometry (DG) guidance laws against a maneuvering and non-maneuvering CM that flies at low altitude and constant speed is described. Simulation results were obtained for two cases: ideal measurements and Kalman filtered noisy line-of-sight angle (azimuth and elevation) and range measurements. Noise tolerance was also examined to determine the best guidance law. For the simulation scenarios, targets are simulated at all aspect angles and for various ranges.

The results show that for a non-maneuvering CM, all three guidance laws give similar results. Against maneuvering targets, DG is better for tail-chase scenarios and PN is better for the forward quadrant of aspect angles. APN performed poorly compared to the other guidance laws examined in these scenarios.

14. SUBJECT TERMS Cruise missile defense, surface-to-air missile, missile guidance, guidance law, proportional navigation, augmented proportional navigation, differential geometry, seeker noise, Kalman filtering.	**15. NUMBER OF PAGES** 149
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

THIS PAGE INTENTIONALLY LEFT BLANK

A STUDY OF THE EFFECTS OF SENSOR NOISE AND GUIDANCE LAWS ON SAM EFFECTIVENESS AGAINST CRUISE MISSILES

Murat DOGEN
First Lieutenant, Turkish Air Force
B.S.E.E., Turkish Air Force Academy, 2008

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2015

Author:	Murat DOGEN
Approved by:	Robert G. Hutchins Thesis Advisor
	Xiaoping Yun Second Reader
	R. Clark Robertson Chair, Department of Electrical Engineering

ABSTRACT

Cruise missiles (CMs) are getting more advanced. To cope with the rapidly growing CM threat, national air-defense systems use surface-to-air missiles (SAMs) as interceptors. To intercept a CM with a SAM, an optimal guidance law should be used. Simulations that represent reality as closely as possible show the effectiveness of the missile system in various scenarios.

A three-degree-of-freedom, discrete-time, and three-dimensional simulation model that compares proportional navigation (PN), augmented proportional navigation (APN), and differential geometry (DG) guidance laws against a maneuvering and non-maneuvering CM that flies at low altitude and constant speed is described. Simulation results were obtained for two cases: ideal measurements and Kalman filtered noisy line-of-sight angle (azimuth and elevation) and range measurements. Noise tolerance was also examined to determine the best guidance law. For the simulation scenarios, targets are simulated at all aspect angles and for various ranges.

The results show that for a non-maneuvering CM, all three guidance laws give similar results. Against maneuvering targets, DG is better for tail-chase scenarios and PN is better for the forward quadrant of aspect angles. APN performed poorly compared to the other guidance laws examined in these scenarios.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AMRAAM	advanced medium-range air-to-air missile
AOA	angle-of-attack
APN	augmented proportional navigation
CM	cruise missile
DG	differential geometry
DOF	degrees of freedom
DSMAC	digital scene-matching area correlation
ECS	engagement control station
EW	electronic warfare
GLONASS	global navigation satellite system
HTPB	hydroxyl-terminated polybutadiene
INS	inertial navigation system
JLENS	Joint Land Attack Cruise Missile Defense Elevated Netted Sensor System
KB	kinematic boundary
LACM	land attack cruise missile
LOS	line-of-sight
NATO	North Atlantic Treaty Organization
NED	North-East-Down
PAC-3	Patriot Advanced Capability-3
PN	proportional navigation
SAM	surface-to-air missile
TERCOM	terrain contour matching

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF SYMBOLS

$\| \|$	magnitude
$\wedge$	estimated value
Δ	discrete time step size
k	discrete time
a_c	commanded missile acceleration
v_c	closing speed
N'	navigational constant
$\dot{\theta}_L$	line-of-sight rate
η_M	missile lead angle
η_T	target lead angle
a_T	target acceleration
$V_T(k)$	target velocity
$x(k)$	state vector
$\hat{x}(k)$	predicted state estimate
$F(k)$	state-transition matrix
$G(k)$	deterministic input-gain matrix
$u(k)$	deterministic input vector
$\upsilon(k)$	process noise
$z(k)$	measurement
$H(k)$	measurement extraction matrix
$w(k)$	measurement noise
$R(k)$	measurement covariance
$Q(k)$	process covariance
$P(k)$	state covariance

$\tilde{z}(k)$	measurement residual
$\hat{z}(k)$	predicted measurement estimate
$W(k)$	filter gain
$S(k)$	innovation covariance
VLA	vertical launch angle
$M(k)$	missile state vector
$F_M(k)$	missile state transition matrix
$M_a(k)$	missile acceleration matrix
$\mathrm{t_{burn}}$	burn-time
T/W	thrust-to-weight ratio
$\mathrm{I_{sp}}$	specific impulse
T	thrust
$\mathrm{mass_{total}}$	mass at launch
$\mathrm{mass_{impact}}$	burnout mass
g	acceleration due to gravity
a_{Mdrag}	missile acceleration due to drag
F_{dp}	parasitic drag force
F_{di}	induced drag force
m	mass
C_{dp}	parasitic drag coefficient
Q	dynamic pressure
S_{REF}	missile cross-sectional area
ρ	density
$v_M(k)$	missile speed
$T(k)$	target state vector
F_{Turn}	target state transition matrix

t_{go}	time-to-go
F_{Turn}	target state-transition matrix for turn
$\theta_{L_{az}}$	line-of-sight azimuth angle
$\theta_{L_{el}}$	line-of-sight elevation angle
$\theta^*_{L_{az}}$	noisy line-of-sight azimuth angle
$\theta^*_{L_{el}}$	noisy line-of-sight elevation angle
σ_{θ_L}	LOS angle sensor standard deviation
n_{rand}	pseudorandom zero mean Gaussian value
r^*	noisy range measurement
σ_r	range sensor standard deviation
f_{noise}	noise multiplier
σ_{r_base}	range sensor baseline standard deviation
$\sigma_{\theta_L\_base}$	LOS angle sensor baseline standard deviation
$q^2(k)$	process noise weighting factor
a_{Tperp}	target acceleration perpendicular to the LOS
a_{Tpara}	target acceleration parallel to the LOS
$a_{Mthrust}$	missile acceleration due to thrust
$a_{u_{perp}}$	portion of the deterministic input perpendicular to the LOS
$a_{u_{para}}$	portion of the deterministic input parallel to the LOS
$R(k)$	LOS vector
$r(k)$	range
$V_M(k)$	missile velocity
$V_c(k)$	closing velocity
$V_r(k)$	LOS velocity vector
$V_p(k)$	perpendicular portion of LOS velocity vector to LOS vector
$n_c(k)$	guidance acceleration command vector

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Cruise missile (CM) technology is a growing threat. CMs can penetrate enemy land without being detected. They are equipped with electronic warfare (EW) countermeasures. They can fly long ranges (i.e., 300–3000 km) with hundreds of kilograms of various types of payload. They have small size that enables them to fit even in a standard 12-meter shipping container [1], and they have low cost compared to other missiles. For all of these reasons, CMs are considered a significant threat. The best solution to manage this threat is to have a surface-to-air missile (SAM) system more advanced than CMs. Simulations that represent reality as closely as possible for this particular scenario help find better guidance laws. This study achieves the following objectives:

- Simulate scenarios of intercept of a CM by a SAM at various ranges in a three-dimensional geometry.
- Examine selected guidance laws without noise against maneuvering and non-maneuvering CM flying at low altitude.
- Find the noise tolerance level for selected guidance laws by simulating scenarios for different ranges.

An intercept of a CM by a SAM launched from ground level and requiring true three-dimensional guidance was simulated. Differing from some previous studies, the commanded guidance acceleration vector is applied to the missile body while the missile is under thrust.

Proportional navigation (PN), augmented proportional navigation (APN), and differential geometry (DG) guidance laws were examined in this thesis. PN implements acceleration on the missile in a direction perpendicular to the missile body to make the LOS (line-of-sight) angle constant, which is equivalent to making the LOS angle rate zero. The functional form of this commanded acceleration is given by [2]

$$a_c = \frac{N' v_c \dot{\theta}_L}{\cos\left(\eta_M\right)} \qquad (1)$$

where N' is the effective navigation ratio and is selected to be five for all the simulations, v_c is the missile-target closing speed, θ_L is the line-of-sight angle (in rad), the

overdot indicates the line-of-sight rate $\dot{\theta}_L$, and the missile lead angle η_M is defined as the angle between missile velocity vector and LOS vector [2].

APN expands the PN guidance law to include the target acceleration as part of the algorithm. The guidance command acceleration perpendicular to the missile velocity vector is

$$a_c = \frac{N' v_c \dot{\theta}_L}{\cos(\eta_M)} + \frac{0.5N' \|a_T\|}{\cos(\eta_M)}. \tag{2}$$

DG guidance differs from PN guidance by including additional commanded acceleration computed from target acceleration $\|a_T\|$ and target lead angle η_T terms. The DG commanded acceleration is [3]

$$a_c = \|a_T\| \frac{\cos(\eta_T)}{\cos(\eta_M)} + \frac{N' v_c \dot{\theta}_L}{\cos(\eta_M)}. \tag{3}$$

This study uses a simplified discrete time, three degrees of freedom model from Osborn [3]. The specifications of the Patriot Advanced Capability-3 (PAC-3) missile were used for the missile model. The target model used Tomahawk cruise missile characteristics to simulate the target's flight altitude, speed, and turn. Both the missile and the target were modeled as a simple point mass in the North-East-Down coordinate system. The target maintained a constant speed at constant altitude. It maintained a straight line trajectory until three seconds before impact. Three-dimensional guidance law implementation requires the guidance command vector, an estimate of LOS angle rate, target and missile lead angles, and an estimate of target acceleration. To obtain these parameters in a noisy environment, the Kalman filtering technique was used.

First, the simulations were run against maneuvering and non-maneuvering targets to see the effect of additional target related terms in the APN and DG guidance laws in a noiseless environment. Next, simulations with Kalman filtered noise were run against maneuvering targets at 2.0, 8.0, and 15.0 km ranges for 20 iterations. Additionally, the maximum noise tolerance to achieve at least a 70% hit probability was tested for PN and DG guidance laws with 50 iterations.

In this study, performance analysis of the guidance laws was based on three parameters: divert, impact time and impact velocity. *Divert* is defined as the integral of magnitude of the commanded guidance acceleration over the entire missile flight time. *Impact time* is desired to be shortest time to intercept the target. Applying less guidance acceleration exposes the missile to less drag, providing a higher *impact velocity*.

In a noiseless environment, all three guidance laws show the same behavior against non-maneuvering targets. All include the same component $N' v_c \dot{\theta}_L$ in their guidance equation, and the target acceleration term has no effect. DG performs better in tail-chase scenarios against maneuvering targets. DG requires less divert and provides higher impact velocity in the aft quadrant up to aspects angles of 76 degrees at 8000 meters initial target range. APN performance cannot compete with PN and DG. The difference in the impact time between PN and DG is insignificant, but APN requires significantly longer impact time.

In the noisy simulations, Gaussian white noise is added to the actual range, LOS azimuth angle, and LOS elevation angle measurements with standard deviations of 10 meters, 1 mrad and 1 mrad, respectively. In the filtered noise study section, only PN and DG were examined because APN showed unsatisfactory behavior in a noiseless environment.

As in the noiseless simulations, DG tends to have better performance in the aft quadrant. Performance increases for DG starts at an aspect angle of ten degrees and continues up to 109 degrees, as shown in Figure 1. In contrast with the noiseless environment results, DG suffers performance degradation between zero and nine degrees of aspect angle (stern chase). Even when the outputs from filters are relatively close to the actual measurements, noisy filter outputs create target accelerations with a magnitude different from zero when the target is not maneuvering. The excessive target acceleration induces extra guidance command and extra drag that slows the missile. The main reason that this excessive guidance command and drag occurs in the first nine degrees is because this scenario has a relatively longer intercept time.

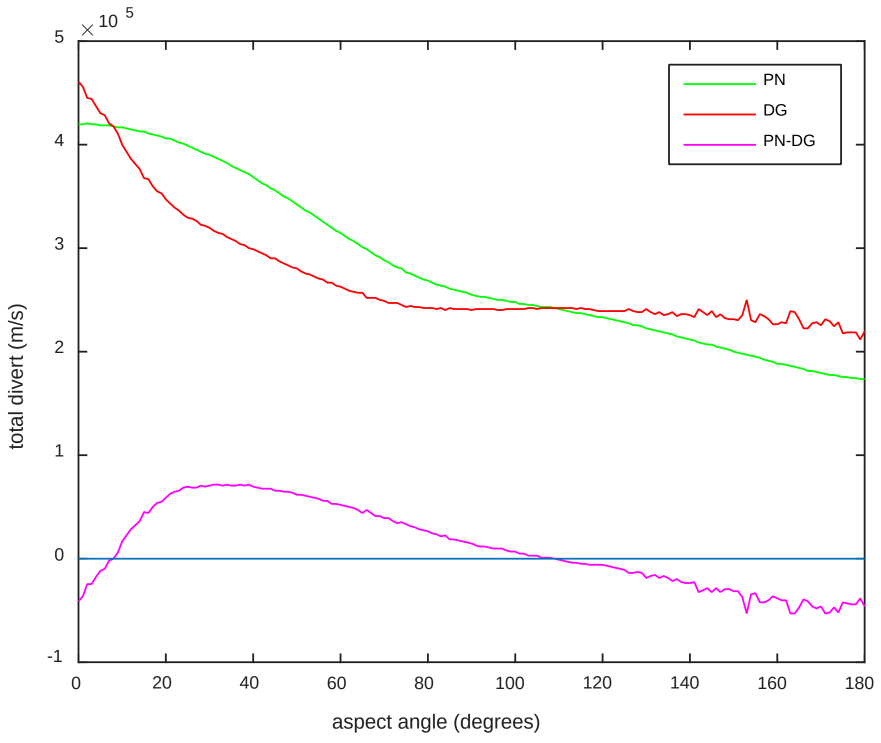


Figure 1. Comparison of divert between PN and DG guidance laws with filtered noise against maneuvering targets at 8000 meters.

As in the results of simulations with noiseless measurements, DG performs better in tail-chase scenarios against maneuvering targets at 8000 meters by intercepting in a shorter time with higher impact velocity.

In the maximum noise factor study, we found DG can tolerate more noise up to 100 degrees and that DG has higher noise tolerance starting from 135 degrees up to 180 degrees at 2000 meters initial target range while maintaining 70% hit probability.

It is important to note that at the simulated ranges all three guidance laws achieved 100 % hit probability against maneuvering and non-maneuvering targets using noiseless measurements. Against non-maneuvering targets, all guidance laws performed the same. Against maneuvering targets, the DG guidance law performed better for tail-chase scenarios under noiseless conditions. In the aft quadrant, DG required less divert and provided a target hit with shorter impact times and higher impact velocity as compared to PN and APN. Although APN hit the target for all ranges considered, performance of APN was inferior and could not compete with APN and DG in these scenarios.

Using the filtered measurements, we saw that DG showed similar behavior in the aft quadrant, requiring less divert, providing shorter impact times, and providing a higher impact velocity. The performance degradation for DG happened for zero to nine degrees of aspect angle because of the noisy estimate of the target acceleration magnitude in its formulation.

Maximum noise factor tests for 2000 meters and 8000 meters showed that DG has higher noise tolerance. DG showed better performance using the analysis parameters in the filtered study part. The higher noise factor values in the aft quadrant for DG match this result. Additionally, DG can tolerate more noise than PN in the forward quadrant.

For future work, more advanced filters, such as an interacting multiple model (IMM), may be used to generate better estimates. A guidance law based on the angle of impact or the predicted intercept point can be examined to determine the tactical performance of a SAM against a CM. A target turn with random initiation time and random acceleration magnitude can be implemented. An aerodynamic model of a missile obtained using software, such as Missile DATCOM, can be implemented in a 6DOF simulation model. Lastly, to decrease the computation time, code generation can be designed to be compatible with parallel computing to run the simulations on a super computer.

LIST OF REFERENCES

[1] D. M. Gormley, "Missile Defense Myopia: Lessons from the Iraq War," *Survival,* vol. 45, no. 4, pp. 61–86, Winter 2003.

[2] P. Zarchan, Ed., *Tactical and Strategic Missile Guidance*, 6th ed. Reston, VA: Amer. Inst. of Aeronautics and Astronautics, 2012.

[3] A. M. Osborn, "A study into the effects of Kalman filtered noise in advanced guidance laws of missile navigation," M.S. thesis, Dept. Elect. and Comput. Eng., Naval Postgraduate School, Monterey, CA, 2014.

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank Professors Robert G. Hutchins, Xiaoping Yun, and Christopher Brophy for their guidance and assistance in this thesis. Special thanks go to Professor Robert G. Hutchins for giving solutions in our long discussions.

I would also like to thank James Calusdian for providing a study area in Controls Laboratory. Lastly, I want to thank Bob Broadston for teaching MATLAB.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

A new technology, the land attack cruise missile, showed its lethality when Germany launched 21,000 cruise missiles (CMs) against Britain and Belgium in World War 2 (WW2). CMs were novel, and although they lacked accuracy, they caused thousands of deaths in WW2. In 1987, 37 sailors died when two Iraqi Exocets accidentally struck the USS *Stark* frigate (FFG-31) while on patrol in the Persian Gulf. During the 2003 Iraq War, U.S. and Kuwaiti Patriot theater ballistic missile defense batteries failed to detect and intercept any of the five modified Iraqi HY-2/CSSC-3 Seersucker CMs launched against Kuwait. One came close to hitting Camp Commando, the U.S. Marine Corps headquarters in Kuwait, on the first day of the war. Another landed just outside a shopping mall in Kuwait City [1]. The inability to detect and intercept hostile CMs in the 2003 Iraq War made the United States change its missile defense systems programs. CM technology is a growing threat. Right now, around 70 countries have CMs in their inventory, and the total approximate number is more than 75,000 CMs. In the next decade, nine countries will be ready to export CMs [1]. Proliferation of this technology without constraints will broaden the CM threat.

The U.S. Department of Defense gives a generic definition for a CM stated as “a guided missile, the major portion of whose flight path to its target is conducted at approximately constant velocity, depends on the dynamic reaction of air for lift and upon propulsion forces to balance drag” [2]. Characteristics of CMs vary significantly. They can be launched from aircraft, ships, submarines, or from the ground. Most CMs use air-breathing engines (i.e., pulsejet, ramjet, turbojet, or turbofan), and some use rocket motors. While some fly at subsonic speeds, advanced CMs can fly up to Mach three. Although there are many variations of CMs, they can be divided in two categories: anti-ship CMs and land-attack cruise missiles (LACM) [3].

Today, CMs enhance the perception of a country’s military power, which is why China and North Korea introduced their new missile systems during national parades.

CMs also play a major role in deterrence against other countries by providing a highly accurate and effective means to penetrate an enemy’s defended areas. If a country possesses a CM that can defeat missile defense systems, then it will limit the adversary’s military and political operations.

CMs can penetrate enemy land without being detected. They are equipped with electronic warfare (EW) counter-measures, can fly long ranges (i.e., 300–3000 km) with hundreds of kilograms of various types of payload, have a small size that enables them to fit even in a standard 12-meter shipping container [4], and have low a cost compared to other missiles. All these reasons make CMs a crucial threat.

State-of-the art CMs can perform very well against current radar and missile systems. Some CMs fly at a high altitude before diving to the target, and some fly at low altitudes to stay below the enemy’s radar horizon. Because of ground clutter, many types of radar used in missile defense systems tend to set their antenna beams above the ground. Also, the range of detection by ground-based radars is limited due to the earth’s curvature. Even airborne surveillance radars strain to detect CMs because of the high level of reflected noise from the ground. Since CMs can update their guidance using Terrain Contour Matching (TERCOM), they can hide behind terrain features to avoid radar systems. With advances in aviation technology, modern CMs have a low radar signature (i.e., radar cross-section). Some also have the ability to deploy chaff or decoys. With this EW counter-measure capability, CMs can deceive and jam radar systems. The combination of all these advancements makes CMs extremely difficult to detect and track.

The high accuracy of guidance systems enables CMs to fly thousands of kilometers to their targets with different types of payloads. Most CMs use high accuracy inertial navigation systems (INS) with navigational updates from the Global Positioning System (GPS) or Global Navigation Satellite System (GLONASS). TERCOM and Digital Scene Matching Area Correlation (DSMAC) systems update navigation to minimize errors. As a result of combining all these navigation systems, the target can be hit to within a few meters range. The Torgos air-launched LACM produced by South

Africa is a good example, because it has a very high accurate hit capability with a circular error probability of two meters.

One of the worst problems is that CMs can carry unidentified munitions, which could be a nuclear warhead or a chemical or biological warhead. For terrorist groups or individuals in third world countries, it is not easy to acquire nuclear weapons; however, they can still produce chemical and biological agents even in a small laboratory. The necessary ingredients for production of these agents can be obtained legitimately since they are already used in many industries. In such cases, the accuracy of the CM is not as important, since the agents will disperse over a vast area with the impact. Also, the contamination may last days depending on weather conditions (rain, wind, etc.).

Another factor that makes CMs dangerous against a missile defense is they can be transported easily without being noticed due to their small size. For example, the Chinese HY-2/CSSC-3 (Seersucker) can fit in and be launched from a shipping container by the internal erector. Another example, the Russian Club-K CM, which works the same way, is shown in Figure 1. The CM disguised in a shipping container can be transported easily by rail, on trucks or ships like regular cargo. When it is transported into close proximity of the target, it is a significant threat.

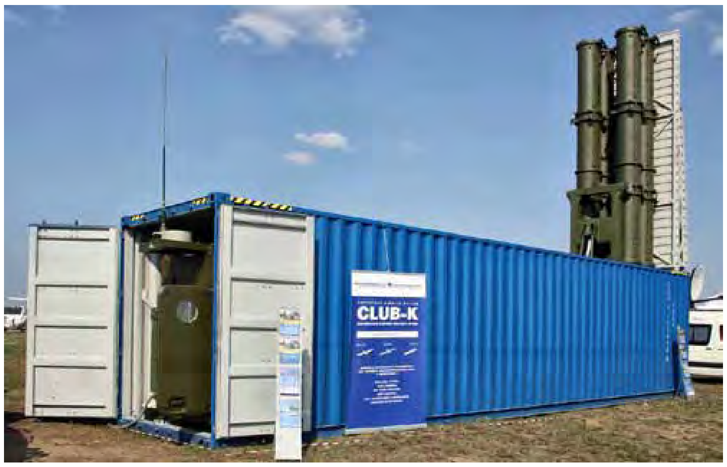


Figure 1. Russian CLUB-K cruise missile can be launched from a standard 12-meter shipping container, from [3].

The low cost of production makes CMs attractive to third world countries since they cannot afford to build the technology from scratch. Compared to ballistic missiles, unmanned aerial vehicles, and fighter aircraft, CMs are cheaper to produce and maintain. This fact sparks the interest of many countries to acquire an arsenal of thousands of CMs.

The technological capabilities of CMs are increasing as well as the number of countries that employ CMs in their arsenals. To cope with this fact, various nations have invested in better missile defense systems. Besides exporting them, countries like Russia and India cooperated to develop the 290 km Brahmos anti-ship CM. As a counter, NATO members (USA, Germany, Netherlands, UK, Denmark Turkey, Romania, Poland, and Spain) contributed to the Maritime Theatre Missile Defence Forum to obtain an impressive and collaborative missile defense. These nine countries allow use of their bases, radars, and missile systems, and share the financial burden with each other [5].

To manage this challenging threat, 15 countries, including the United States, use the latest version of the Patriot Advanced Capability-3 (PAC-3) missile system. It is combat-proven, highly agile, and the first hit-to-kill interceptor. Besides using the hit-to-kill mechanism to destroy the target, it can also use the 73 kg fragmentation warhead that works with a proximity fuse. This is especially designed to intercept air-breathing targets such as CMs and aircraft [6].

The PAC-3 missile system consists of a phased array radar set, an engagement control station (ECS), a battery command post, an electric power plant, an antenna mast group, a communications relay group, and launching stations, as shown in Figure 2. The antenna mast group and the communications relay group provide 30 kilometers of remote launch capability. Additionally, target acquisition from integrated radar systems increases the defended area.

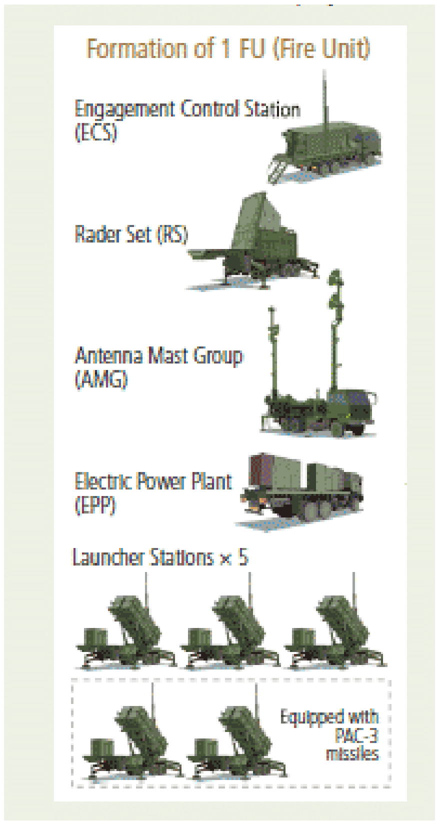

Formation of 1 FU (Fire Unit)
Engagement Control Station (ECS)
Rader Set (RS)
Antenna Mast Group (AMG)
Electric Power Plant (EPP)
Launcher Stations × 5
Equipped with PAC-3 missiles

Figure 2. Formation of a PAC-3 Fire unit, after [7].

The missile uses a solid propellant (hydroxyl-terminated polybutadiene) rocket motor, 180 mini-solid-propellant, altitude-control thruster motors and inertial guidance to navigate to an intercept point provided at launch from the phased array radar. A few seconds before the intercept time, the highly-accurate onboard active radar seeker turns active in the terminal homing phase to intercept any highly maneuverable CM.

The PAC-3 missile system was first deployed in 1995, and there have been many upgrades to the software, launchers, and radar. Integration of the PAC-3 missile system with other defense and communication systems improved its interoperability. There are ongoing development projects to enhance its capability. In the last few years, there have been many flight tests where PAC-3 demonstrated that it could intercept cruise missiles. Chronologically, the PAC-3 missile succeeded in two intercepts of low flying MQM-107 Streaker target drones in July 2000 [6], an intercept of a BQM-74 target drone that simulates advanced CM flying at a very low altitude in a cluttered background in October

2001 [6], an intercept of a ballistic missile and MQM-107 Streaker target drone as a surrogate CM in a ripple fire in September 2004 [8], two intercepts of low-flying air-breathing targets in July 2007 [9], and an intercept of a low-flying CM surrogate drone picked up by the Joint Land Attack Cruise Missile Defense Elevated Netted Sensor System (JLENS) surveillance radar in April 2012 [10]. The growing threat requires the PAC-3 missile system to improve continuously. There will be new firing tests as software and hardware are upgraded.

Figure 3. PAC-3 intercept of a low-level target, from [6]

CMs are improving, and the threat level is increasing as CMs are acquired by many countries and as they become more advanced. Hence, a country needs to have more advanced missile systems to stay ahead against the improvements in CMs. One of the most important parts of a missile is the guidance part. A guidance law is defined as an algorithm that determines the required commanded missile acceleration [11]. If we use the best guidance law, other missile parameters such as length, mass, operational range, kill probability, intercept time, and so forth are also affected. For this reason, guidance laws should be tested to improve performance against such a challenging threat.

CMs are cheap and easily acquired by our adversaries. Proliferation and technological capabilities of CMs are increasing over time. The best solution to manage this threat is to have a surface-to-air missile (SAM) system more advanced than CMs. Simulations that represent reality as closely as possible for particular scenarios help find a better guidance laws.

A. PREVIOUS STUDIES

Among the many guidance law studies, three [12]-[14] were chosen as primary sources for this study. Previous studies suggest that proportional navigation (PN) is only good for non-maneuvering targets. Another result says that a guidance law that needs an input of target acceleration, such as augmented proportional navigation (APN) and differential geometry (DG), show improvement against high acceleration maneuvering targets [12]-[14].

In his thesis, Broadston [12] studied five different guidance laws: proportional navigation, velocity compensated proportional navigation, bang-bang, differential games, and augmented proportional navigation. He designed a Simulink™ Model to simulate the three axes of translation in a North-East-Down (NED) coordinate system and three axes of rotation. These axes are the six degrees of freedom. His study examined the guidance laws in three scenarios: non-maneuvering CM at 50 meters altitude, a maneuvering target at a constant altitude of 6000 meters, and a non-maneuvering target at a constant altitude of 6000 meters. In the maneuvering target scenario, three seconds before the intercept by the missile, the target initiated an evasive constant 6-g turn toward the missile.

Broadston chose the kinematic boundary (KB) methodology to compare these guidance laws, taking PN as the baseline. He defined the KB as the maximum range at which a missile can reach the kill radius of the target in a noiseless environment [12]. He assumed a kill radius of five meters for the Advanced Medium-Range Air-to-Air Missile

(AMRAAM). KB is a good tool to measure the effectiveness of a missile, but it is not sufficient by itself in optimal guidance law studies.

In his noise study, Broadston added noise to range, closing speed, line-of-sight (LOS) angle, and LOS rate; however, modern radar and seeker systems usually measure range and LOS angle and compute closing speed and LOS angle rate from the measurements. With this noise model, he tested the guidance laws only at a single aspect angle. By contrast, the performance of guidance laws at many aspect angles is examined while noise is present in this study.

At first, Broadston examined the effect of gain (N') for PN. He achieved the best overall results with $N' = 5$, and he chose PN with $N' = 5$ as his baseline for all other simulations. His results showed that only APN improved the KB [12]. For this reason, APN was included in this study. Also, to better understand the comparison between the guidance laws, the noise scenarios are run at each aspect angle, unlike Broadston, who did it for only one aspect angle.

Since APN and PN offered the best results in earlier studies, in 2011 Pehr [13] continued Broadston's work and studied these two guidance laws as well as the DG guidance law. He also used KB to measure the effectiveness of these three guidance laws. He used a 6DOF Simulink model, but he modified the drag model of the missile. Similarly, he took $N' = 5$ as baseline since it gave better results in his model. As expected, all three guidance laws gave similar results against a non-maneuvering target in a noiseless environment.

In his noise study, Pehr's goal was to see the degradation level caused by unfiltered noise. For a non-maneuvering target when a baseline noise was added, the degradation happened for APN and DG because added terms in the guidance law made them more complex. When unfiltered noise was added in a maneuvering target scenario, DG gave the best performance due to the added target acceleration term in the guidance law. DG performed better in both maneuvering and non-maneuvering target scenarios [13].

Osborn did a follow-on study in 2014 and used a simplified aerodynamic model for the AIM-120 AMRAAM missile. He performed his simulations using a three degree of freedom (3DOF) model built in MATLAB. His main goal was to compare the effects of Kalman filtered noise for PN and DG. As in the previous studies, he used KB methodology for his comparisons [14].

Osborn’s noise study was better since he studied Kalman filtered noise instead of a “direct insertion of noise method” like Broadston and Pehr did [11]. Osborn’s study showed that KB degraded more when DG was used. On the other hand, DG had better performance for tail-chase scenarios. With Kalman filtering, he examined the maximum noise tolerance level for both guidance laws. In tail-chase scenarios, noise tolerance was higher for DG; for head-on scenarios, PN had better noise tolerance. Also, his results showed that PN was more susceptible to noise [14].

It is clear that each study contributes to and improves on the previous one. It is observed that DG has better performance in a noiseless environment. On the other hand, because of the target acceleration term, DG and APN are more susceptible to noise in two-dimensional, air-to-air engagement scenarios.

B. METHODS

Many guidance law studies simulate the flight characteristic at a constant altitude. An intercept of a CM by a SAM launched from ground level requires true three-dimensional guidance and is simulated in this study. Consequently, guidance laws are compared against both maneuvering and non-maneuvering targets in a noiseless environment as well as an environment with significant measurement noise in this study.

Differing from some previous studies, we apply the commanded guidance acceleration vector to the missile body while the missile is still under thrust. This is important to succeed for short range target engagement scenarios.

A modified 3DOF, discrete time simulation model similar to a previous thesis study is used here [14]. Since DG needs an estimate of target acceleration as input, this guidance law is more susceptible to sensor noise. Using Kalman filtering methodology, we examine sensor noise effects on guidance laws for the specified scenarios.

KB is one of the methods used in optimal guidance law studies to measure the effectiveness of guidance laws but by itself does not give complete information. This study uses a missile model that simulates the PAC-3, which has a designated engagement range between 2 and 15 km. For this reason, instead of the KB method, other parameters are examined to compare the three guidance laws. These parameters are divert-of-guidance acceleration command, time-of-intercept, and missile-terminal speed. Also, for a better performance analysis, intercept scenarios are examined for many aspect angles, not just one aspect angle as Broadston did.

Although examining a specific SAM launched from ground level trying to intercept a CM flying at low altitude is a completely different case from these other studies, some results from Broadston's, Pehr's and Osborn's studies were used as guidelines.

C. OBJECTIVES

Previous studies suggest that PN is best for non-maneuvering targets and guidance laws that need an input of target acceleration—such as APN and DG— show improvement against high acceleration maneuvering targets. In most of the previous simulations, intercepts are assumed to happen at a constant altitude. An intercept of a Tomahawk missile by a PAC-3 missile system is simulated in this study.

The following objectives were achieved in this study:

- Intercept of a CM by a SAM at various ranges in a three-dimensional geometry are simulated.
- Guidance laws without noise against maneuvering and non-maneuvering CM flying at low altitude are examined.
- The noise tolerance levels for selected guidance laws by simulating scenarios at different ranges are found.

D. THESIS ORGANIZATION

This thesis is organized as follows. An overview of PN, APN, and DG guidance laws is provided in Chapter II. The Kalman filtering algorithm is overviewed in Chapter III. The 3DOF methodology including missile model, target model, drag model and filtering modifications is described in Chapter IV, as well as the assumptions and validations for the missile thrust model and missile drag model. Simulation results and performance analyses for different scenarios for both noisy and noiseless environments are contained in Chapter V. The results and recommendations for further research are discussed in Chapter VI. The MATLAB code used in this study is contained in the Appendix.

THIS PAGE INTENTIONALLY LEFT BLANK

II. GUIDANCE LAWS

An overview of the guidance laws examined in this thesis is provided in this chapter. The guidance laws considered are proportional navigation (PN) [15]–[16], augmented proportional navigation (APN) [15], and differential geometry (DG) [17]. Most guided missile systems use PN or some derivative [15]–[17]. PN by itself often performs unsatisfactorily against highly maneuverable targets [11]. Because of this fact, Broadston [12] explored a variety of hybrid guidance laws, including APN. He found all but APN were less satisfactory than PN when implemented on missiles subject to drag and other aerodynamic effects. Pehr [13] expanded on these results by exploring DG and found it to be a viable guidance law, outperforming both PN and APN in certain applications. Osborn [14] explored these guidance laws using Kalman filtering techniques in the presence of noisy measurements. He concluded that DG has better performance for tail-chase scenarios in a noisy environment [14].

Broadston, Pehr, and Osborn produced their results using primarily two-dimensional guidance. This technique is inadequate to explore the problem of intercepting low-flying cruise missiles using ground-launched surface-to-air missile (SAM) interceptors. This more complex situation in the presence of noise is explored in this thesis using Kalman filtering technique. A mathematical description of the three guidance laws is given in the following section.

A. PROPORTIONAL NAVIGATION

The “constant bearing rule” states that when two objects move in straight lines at constant but potentially different speeds, they will collide. This result happens due to the constant line-of-sight (LOS) bearing angle between the two objects. This rule is based on physical law and simple geometry.

PN implements the constant bearing rule. The rule requires the LOS angle to be constant, which is equivalent to requiring the LOS angle rate to be zero; hence, PN implements acceleration on the missile in a direction perpendicular to the LOS vector.

This acceleration is proportional to the LOS angle rate and the closing speed, which drives the LOS rate toward zero [16]. The functional form of this commanded acceleration is given by [15]

$$a_c = N' v_c \dot{\theta}_L \tag{2.1}$$

where N' is a unitless designer-chosen gain (usually in the range of three to five), also known as the effective navigation ratio, v_c is the missile-target closing speed, and θ_L is the line-of-sight angle (in rad). The overdot indicates the time derivative of the line-of-sight angle, that is, the line-of-sight rate $\dot{\theta}_L$ [15].

When the guidance acceleration is applied perpendicular to the LOS vector, it is called "true" PN [18]. Velocity-referenced guidance laws are more reasonable in practical implementation. Because of this, in the simulation described here, the guidance acceleration command is applied perpendicular to the missile body (perpendicular to the missile velocity vector). This type of implementation is called the "pure" PN [18] guidance law. Since this thesis is a follow-up study after [13], [14], the pure PN form is used; hence, the magnitude of the guidance command acceleration is

$$a_c = \frac{N' v_c \dot{\theta}_L}{\cos(\eta_M)} \tag{2.2}$$

where η_M is the missile lead angle, which is defined as the angle between missile velocity vector and the LOS vector, and is depicted in Figure 4. After Broadston examined the effect of gain N', he recommended using $N' = 5$ [12]; therefore, in this study the navigational constant N' was selected to be five for the all simulations, as was also done by Pehr and Osborn. The effect of the term in the denominator is to give the correct acceleration magnitude in the direction perpendicular to the missile velocity vector [18], [19].

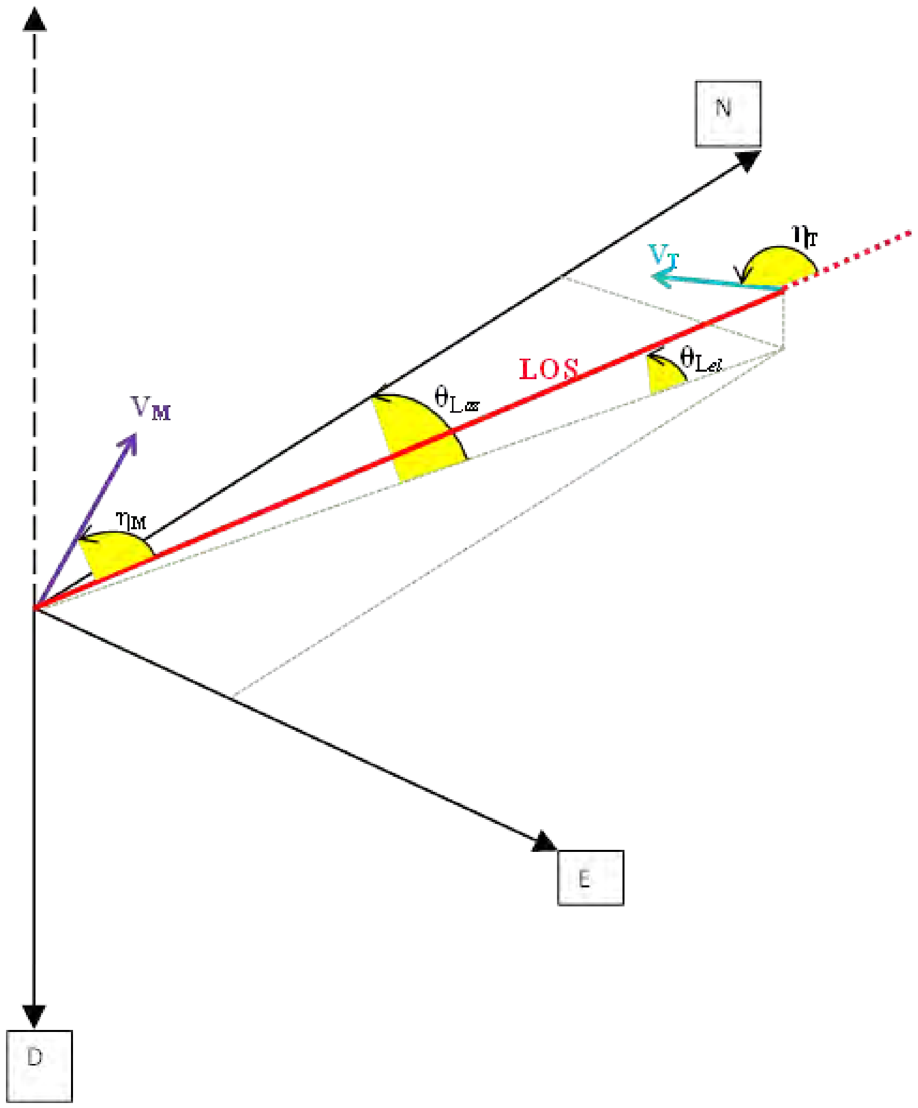


Figure 4. Three-dimensional encounter geometry in NED coordinate system.

B. AUGMENTED PROPORTIONAL NAVIGATION (APN)

Lateral divert is directly related to the amount of fuel required by the interceptor to implement the guidance law. The lateral divert is the total area under the absolute value of the acceleration curve. Against accelerating targets, APN is a more fuel-efficient guidance law than the PN guidance law; this is because APN has one-half the lateral divert requirements of PN regardless of the effective navigation ratio [15]. This performance is achieved because APN expands the PN guidance law to include the target acceleration $\|a_T\|$ as part of the algorithm. The guidance command acceleration presented in [15] is

$$a_c = N'v_c\dot{\theta}_L + 0.5N'\|a_T\|. \tag{2.3}$$

To apply the guidance command acceleration perpendicular to the missile velocity vector, (2.3) is modified to:

$$a_c = \frac{N'v_c\dot{\theta}_L}{\cos(\eta_M)} + \frac{0.5N'\|a_T\|}{\cos(\eta_M)}. \tag{2.4}$$

C. DIFFERENTIAL GEOMETRY

The use of differential geometric concepts gives the approach a sound basis for more generalized guidance techniques. This allows for curved and straight trajectories to be considered for both the target and the missile [17].

In [13] and [14], the derivation of the DG guidance law was presented. From this derivation, the guidance command acceleration applied perpendicular to the missile velocity vector is [14]

$$a_c = \|a_T\|\frac{\cos(\eta_T)}{\cos(\eta_M)} + \frac{N'v_c\dot{\theta}_L}{\cos(\eta_M)}. \tag{2.5}$$

The lead angles of the target and the missile are denoted by η_T and η_M, respectively. These angles are shown in Figure 4 as the angles between velocity vectors with respect to the LOS vector. The magnitude of the target acceleration $\|a_T\|$ and the target lead angle η_T represent the curvature of the target maneuver.

DG guidance modifies the pure PN guidance law expressed in (2.1) and differs from PN guidance by including additional commanded acceleration computed from target acceleration $\|a_T\|$ and target lead angle η_T terms. As with APN, the target acceleration vector is obtained from the LOS angles (azimuth and elevation) and range measurements. In the noiseless simulation, the target acceleration estimate is obtained by $\|a_T\| = \|\Delta\mathrm{V}_\mathrm{T} / \Delta t\|$, that is, the target velocity change for one time step Δ. When the noise is applied, the target acceleration estimate is obtained from the Kalman filtered output parameters.

III. KALMAN FILTERING

The Kalman filter provides an optimal and efficient computational procedure to estimate a multi-dimensional state vector. To obtain the necessary parameters to implement guidance laws, we need the estimates of LOS angles and range as well as the estimates of their first and second time derivatives. Like Osborn's research, this study uses the Kalman filtering technique to estimate these parameters from the noisy measurements which are obtained from the onboard sensors. An overview to the Kalman filtering algorithm, including the dynamic plant model, the prediction phase, and the correction phase, is provided in this chapter.

A. DYNAMIC PLANT MODEL

The dynamic plant model is constructed in the form of a linear, dynamic state space representation to implement the Kalman filter. The plant model consists of the dynamic state model and the measurement model. The state model is represented in discrete time form by the state equation

$$x(k+1) = F(k)x(k) + G(k)u(k) + \upsilon(k), \tag{3.1}$$

where $F(k)$ is the state-transition matrix, which translates the state vector to the next time step $k+1$. A standard state-transition matrix for a state vector with three components (e.g., $r, \dot{r}, \ddot{r}$) is

$$F = \begin{bmatrix} 1 & \Delta & \Delta^2/2 \\ 0 & 1 & \Delta \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.2}$$

Here, the discrete time step size Δ is the time difference between two time steps k and $k+1$, and $u(k)$ is defined as the deterministic input vector and accounts for state variations that are already known. The input-gain matrix $G(k)$ is a gain matrix and is used to weigh $u(k)$. The process noise $\upsilon(k)$ accounts for unmodeled, random inputs. For a three-dimensional dynamic plant, the output of the state equation at time $k+1$ is the state vector $x(k+1)$, which consists of a parameter of interest such as range $r(k+1)$ and the parameter's first and second time derivatives. Hence, the state vector is defined as

$$x(k+1)=\begin{bmatrix} r(k+1) \\ \dot{r}(k+1) \\ \ddot{r}(k+1) \end{bmatrix}. \tag{3.3}$$

The measurement model at time $k+1$ is [20]

$$z(k+1)=H(k+1)x(k+1)+w(k+1). \tag{3.4}$$

$H(k+1)$ is the measurement output matrix and is defined as

$$H=\begin{bmatrix}1 & 0 & 0\end{bmatrix}. \tag{3.5}$$

The measurement noise $w(k+1)$ accounts for the sensor errors [19]. Both process noise and measurement noise are modeled as zero mean, white Gaussian noise in the Kalman filtering technique. The vectors $w(k)$ and $\upsilon(k)$ are assumed to be uncorrelated.

The measurement covariance $R(k)$ is the covariance of the measurement noise vector $w(k)$; $R(k)$ relates to sensor accuracy and depends on the standard deviation σ of the measurements [19]:

$$R(k)=\left[\sigma^2\right]. \tag{3.6}$$

The matrix $Q(k)$ is the covariance of the process noise $\upsilon(k)$. It is used to account for the expected state variation between measurements. A large process covariance makes the filter place greater emphasis on the latest measurement in order to adapt quickly to large changes in the measurements. On the other hand, a filter with a small process covariance pays more attention to the averaging of previous measurements. Using a small process covariance produces a better estimate for a motion where no large state variations are expected [14], [21]. The matrix $P(k)$ is the covariance of the state and is a measure of the accuracy of the state estimate. The initial value for the state covariance $P(0)$ is computed from $R(k)$.

The state vector, state-covariance matrix $P(k)$, deterministic input vector $u(k)$, measurement covariance $R(k)$, and process noise covariance matrix $Q(k)$ are presented in detail in Chapter IV for range and bearing filters separately.

B. PREDICTION PHASE

The prediction phase is the first portion of the Kalman filtering algorithm. The filter inputs in the prediction phase are the corrected state estimate, state covariance, and deterministic input vector from the current time k. The outputs of this phase are the predicted state estimate, predicted state covariance, and predicted measurement estimate for time $k+1$ [21]. These outputs are used as inputs for the correction phase.

The predicted state estimate $\hat{x}(k+1|k)$ is derived from the translation of the corrected state estimate, taking the deterministic input into account. The predicted state estimate is given in the state-prediction equation as [20]

$$\hat{x}(k+1|k) = F(k)\hat{x}(k|k) + G(k)u(k). \tag{3.7}$$

The predicted state-estimate covariance $P(k+1|k)$, which reflects the accuracy of the predicted state estimate, is calculated from the corrected state covariance and process covariance of the previous time step k. The addition of the process noise covariance $Q(k)$ allows the predicted state estimate covariance to account for the unpredictable state variation from time k to $k+1$. The predicted state estimate covariance $P(k+1|k)$ is obtained from [20]

$$P(k+1|k) = F(k)P(k|k)F(k)' + Q(k) \tag{3.8}$$

where $F(k)'$ is defined as the inverse of state-transition matrix $F(k)$. The measurement output matrix gives the predicted measurement estimate from the predicted state estimate. Predicted measurement $\hat{z}(k+1|k)$ is [20]

$$\hat{z}(k+1|k) = H(k+1)\hat{x}(k+1|k). \tag{3.9}$$

C. CORRECTION PHASE

The correction phase updates the state estimate and covariance from the prediction phase. The last part of the Kalman filter cycle is complete when the corrected state estimate and covariance are generated. Measurement residual and the filter gain update the predicted state estimate. The corrected state estimate for time $k+1$ is

$$\hat{x}\left(k+1\middle|k+1\right)=\hat{x}\left(k+1\middle|k\right)+W\left(k+1\right)\tilde{z}\left(k+1\right). \tag{3.10}$$

The vector $\tilde{z}\left(k+1\right)$ is the measurement residual, which is also called innovation and can be defined as the difference between the measurement and the predicted measurement estimate; that is, [20]

$$\tilde{z}\left(k+1\right)=z\left(k+1\right)-\hat{z}\left(k+1\middle|k\right). \tag{3.11}$$

The covariance of the measurement residual, which is also called the innovation covariance $S\left(k+1\right)$, is used to find the filter gain $W\left(k\right)$. The innovation covariance $S\left(k+1\right)$ is defined as [20]

$$S\left(k+1\right)=R\left(k+1\right)+H\left(k+1\right)P\left(k+1\middle|k\right)H\left(k+1\right)'. \tag{3.12}$$

The filter gain $W\left(k\right)$, which is used to find the corrected state estimate and corrected state estimate covariance, is specified as [20]

$$W\left(k+1\right)=P\left(k+1\middle|k\right)H\left(k+1\right)'S\left(k+1\right)^{-1}. \tag{3.13}$$

With a small gain, the state estimate relies more on the previous measurements, while a large gain makes the filter rely on to the current measurement more.

The filtering cycle ends by calculating the corrected state-estimate covariance $P\left(k+1\middle|k+1\right)$. The corrected state covariance can be found from [20]

$$P\left(k+1\middle|k+1\right)=P\left(k+1\middle|k\right)-W\left(k+1\right)S\left(k+1\right)W\left(k+1\right)' \tag{3.14}$$

or by the Joseph form, which is given by

$$P\left(k+1|k+1\right)=\left[I-W\left(k+1\right)H\left(k+1\right)\right]P\left(k+1|k\right)\left[I-W\left(k+1\right)H\left(k+1\right)\right]'\ldots$$
$$+W\left(k+1\right)R\left(k+1\right)W\left(k+1\right)'. \tag{3.15}$$

Derivations for the corrected state-estimate covariance in (3.14) and (3.15) are equivalent algebraically, but the Joseph form shown in (3.15) is more stable compared to (3.14). Therefore, it is recommended that the Joseph form be used to find the corrected state covariance, even it requires more computations [20], [21].

D. SUMMARY

The Kalman filter algorithm, which mitigates sensor errors and provides more accurate estimates of the state vectors, was the focus of this chapter. The state and measurement models provide the state equations for implementing a linear Kalman filter on each of the required parameters, the LOS angles and range. The prediction phase gives the predicted state estimate, predicted state covariance, and predicted measurement estimate for time $k+1$. The correction phase uses the predicted estimates and measurement to end a cycle of each Kalman filter by generating a corrected state estimate and covariance for time $k+1$. The general Kalman filter algorithm is tuned for LOS angles and range by setting different process noise covariance matrices and different deterministic inputs. In Chapter IV, we examine the 3DOF simulation methodology, including modifications to the Kalman filter.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. THREE DOF SIMULATION METHODOLOGY

The simulation methods used in this thesis are described in this chapter. A simplified discrete time 3DOF model from Osborn is used [14]. It is easier to examine the influence of guidance laws on the missile's performance using a 3DOF model because it contains significantly fewer parameters than the continuous time 6DOF model used by Broadston [12]. The 3DOF model is achieved by treating both the missile and the target as point masses in the coordinate system.

The simulated missile and target models are presented in this chapter. The specifications of the PAC-3 missile are used to design motion, thrust, and drag models of the simulated missile. The target model uses Tomahawk cruise missile characteristics to simulate target's flight altitude, speed, and turn. The noise model and Kalman filter design are also explained in the remaining sections. The derivation of the required parameters for three-dimensional guidance laws for both noiseless simulations and the simulations using filtered noisy measurements are given in the last section.

In this simulation, the missile is modeled in the North-East-Down (NED) coordinate system, which is commonly used for tactical missiles. The NED coordinate system usually has the origin situated in the Earth's tangent plane directly below the missile. The *x*-axis points north and the *y*-axis points east. To agree with the right-hand rule, the *z*-axis points vertically downward [11]. The Flat Earth assumption is made, and the angular velocity of earth's rotation is neglected because the simulated missile's maximum range is only 15 km.

Simulations were run at the specified range and target aspect angle. The aspect angle was incremented from zero degrees to 180 degrees, and this process was repeated for each guidance law. Either the target was hit or missed; the simulation stopped when any of the following conditions occurred:

- The closing speed becomes negative.
- A hit occurs or the range is less than 5.0 meters.
- The altitude goes below ground level.

Additional conditions were added to the closing speed requirement for the following purposes:

- to prevent simulation stop before the guidance law was applied
- to let the missile gain speed ($V_M > V_T$)

After the guidance law was applied, the program stopped when the closing speed became negative. At that time, the separation between the missile and target is a minimum.

Vectors or matrices related to missile and target are represented with M and T subscripts, respectively. For example, $z_M(k)$ is the position of the missile on *z*-axis at time k. A dot above a symbol indicates a time derivative; two dots, a second derivative. For example, $\dot{x}_T(k)$ is the target speed on *x*-axis at time k.

A. MISSILE MODEL

In the simulations, the PAC-3 missile characteristics are used to design the missile model. The specifications of the PAC-3 missile and assumptions for missile motion, thrust, and drag are described in this section. Various forces act on the missile body; therefore, the missile model must account for the effects of engine thrust, parasitic and induced drag forces, the earth's gravitational force, and the guidance command acceleration.

Guidance acceleration is limited so as not to exceed the missile's lateral acceleration capability. For short range target engagement scenarios, the missile is still under thrust when the guidance acceleration command is applied. For this reason, it is important to use a commanded acceleration limiter. Broadston and Pehr set a 30 g acceleration limit for each dimension of the commanded guidance acceleration [12], [13]. Osborn set the limit to 50 g for PN and 15 g for DG guidance in the *x-y* plane [14]. Our commanded acceleration is also applied in three dimensions as was done in Broadston's and Pehr's studies [12], [13]. To prevent the simulated missile's acceleration capability from being exceeded, a maximum of 50 g guidance acceleration is applied to the missile.

To model the simulated missile's motion, thrust, and drag, some assumptions were made. These assumptions depend on the missile's specifications as obtained from open source literature. The specifications of the PAC-3 missile are shown in Table 1. It should be noted that not all of these parameters are required for the 3DOF model, and the guidance laws used in this study are not derived from those used on the PAC-3 missile.

Table 1. Missile model parameters using specifications of the PAC-3 missile, after [6].

Specifications of the PAC-3 missile	
Length	overall: 5.205 m (17 ft 1 in)
Diameter	255 mm (10.04 in)
Weight	launch: 315 kg (694 lb) impact: 142 kg (313 lb)
Performance Speed	speed: 3,305 kt (6,120 km/h; 3,803 mph; 1,700 m/s)
Range	8.1 n miles (15 km; 9.3 miles)
Warhead:	kinetic (back-up HE fragmentation enhanced warhead)
Guidance:	INS, mid-course update, active radar (Ka-band)
Propulsion type:	solid propellant (175 kg of HTPB propellant) (Atlantic Research composite rocket motor with special attitude control section for in-flight maneuvering)

1. Missile Motion

At time zero, the missile is located at the origin. It is assumed in the following model development that the heading error is initially zero. Before launch, the missile is directly pointing at the target. The target is initialized on the positive *x*-axis, separated from the missile by the range being tested, which is 2 to 15 km. The initial heading of the target, measured from the *x*-axis, is set to the desired aspect angle. A vertical launch angle also needs to be computed because this is a simulation of a SAM launched at ground level; therefore, the vertical launch angle is set between 15 and 41 degrees. The vertical launch angle VLA in degrees is computed using

$$\mathrm{VLA}=15 + 2(\mathrm{D}_{\mathrm{target}} - 2\mathrm{km}) \tag{4.1}$$

where $\mathrm{D}_{\mathrm{target}}$ is the initial range to the target in km. The missile state vector with position and velocity components along the *x*, *y*, and *z* axes is used to calculate missile motion for each time step. The missile state vector $M(k)$ at any given discrete time step k is

$$M(k)=\begin{bmatrix} x_M(k) \\ \dot{x}_M(k) \\ y_M(k) \\ \dot{y}_M(k) \\ z_M(k) \\ \dot{z}_M(k) \end{bmatrix} \tag{4.2}$$

where the missile state vector at the next time step $M(k+1)$ is computed using a transition matrix and missile acceleration matrix and is specified by

$$M(k+1)=F_M M(k)+M_a(k)$$
$$=\begin{bmatrix} 1 & \Delta & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \Delta & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x_M(k) \\ \dot{x}_M(k) \\ y_M(k) \\ \dot{y}_M(k) \\ z_M(k) \\ \dot{z}_M(k) \end{bmatrix}+\begin{bmatrix} \frac{\ddot{x}_M(k)\Delta^2}{2} \\ \ddot{x}_M(k)\Delta \\ \frac{\ddot{y}_M(k)\Delta^2}{2} \\ \ddot{y}_M(k)\Delta \\ \frac{\ddot{z}_M(k)\Delta^2}{2} \\ \ddot{z}_M(k)\Delta \end{bmatrix}. \tag{4.3}$$

The missile transition matrix F_M translates the state vector from time k to $k+1$ by adding displacement in *x*, *y*, and *z* axes due to the velocity. The missile acceleration vector $M_a(k)$ is used to define missile displacement due to thrust, drag, gravity, and commanded guidance.

2. Thrust Characteristics

Designing the thrust model of the missile is one of the most important parts of this simulation. A realistic thrust model requires the following parameters: missile burn-time t_{burn}, thrust-to-weight ratio (T/W), and specific impulse (I_{sp}). These parameters are estimated since there is not any open source reference that provides specific information about the thrust characteristics of the PAC-3 missile. The T/W ratio of the MIM-104A PAC-2 missile, which is 15.57, is used also for the PAC-3 missile [22]. The missile has a single stage motor with a solid propellant of hydroxyl-terminated polybutadiene (HTPB). The I_{sp} value for the HTPB propellant motor is between 260 and 265 seconds [23] and chosen to be 260 seconds for this particular model. Given the assumptions for T/W and I_{sp} parameters, the required missile thrust (T) and burn-time t_{burn} parameters, respectively, are calculated using

$$T = (T/W)\text{mass}_{total}g \tag{4.4}$$

and

$$t_{burn} = \frac{I_{sp}\ (\text{mass}_{total} - \text{mass}_{impact})g}{T} \tag{4.5}$$

where mass_{total} is missile's mass at the launch, mass_{impact} is burnout mass, and g is earth's gravitational force. As a result, for the "boost phase" of flight, the computed values for T and t_{burn} are 48,087 Newton and 9.1711 seconds, respectively.

Propellant flow rate and exhaust velocity are assumed to be constant to give constant thrust force. As a result the missile accelerates at an increasing rate. This rate increase occurs because the missile's overall mass decreases as propellant is burned [11]; this is shown in Figure 5, which gives the thrust profile of the simulated missile.

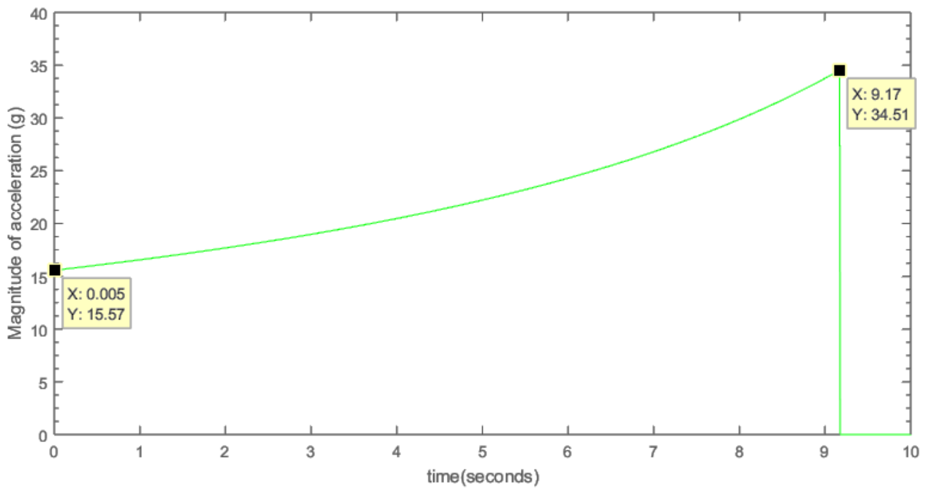


Figure 5. As the propellant burns, the thrust acceleration increases for the simulated PAC-3 missile.

3. Drag Model

Thrust and lift are the main forces that make the missile fly and come with a penalty. Lift and velocity cause a drag force on the missile. Total drag can be broken down into three major components: parasitic drag, induced drag, and wave drag. Parasitic drag includes skin friction drag, form drag, and interference drag. Interference drag results from vortices created at the intersection of surfaces and is neglected in this study. Induced drag happens due to the downwash of wings. Wave drag is also called supersonic drag. To simplify the simulation, the drag model is designed with the first two components, parasitic and induced drag, considered.

While parasitic drag depends on the missile shape and the cross-sectional area when the missile is at level flight, induced drag is caused by the commanded guidance acceleration. For the mass m of the missile at a given time, the total drag force is calculated by adding induced and parasitic drag forces. The total drag force is then converted to an acceleration vector and applied in the opposite direction of the missile velocity vector. The magnitude of acceleration due to drag becomes

$$\left\| a_{Mdrag} \right\| = \frac{\left(F_{dp} + F_{di} \right)}{m} \tag{4.6}$$

where F_{dp} is defined as parasitic drag force and F_{di} is defined as induced drag force.

a. Parasitic Drag

Movement of the missile in the airstream causes the air to move over the missile surface and causes turbulence. The drag force defined by the friction on the missile surface is called parasitic drag. It consists of three types of drag: skin friction drag, form drag, and interference drag. Skin friction drag is caused by the friction on the missile surface that contacts the atmosphere [24]. Form drag is related to the missile's shape. Interference drag results from the overlapping of airstreams at the intersection of the wing and the fuselage. In this simulation, interference drag is not included.

In this study, Osborn's parasitic drag model was used because his model is very close to the empirical drag model [19], which clearly shows that at transonic speed the missile has the highest drag coefficient. The parasitic drag coefficient C_{dp} as a function of Mach number is shown in Figure 6, where two separate curves show the boost and coast phases of flight [14]. In an air vehicle with a small wing area (e.g., high-speed missile), the fuselage cross-sectional area (normal to the flow) is often considered as the reference area. Parasitic drag force F_{dp} is proportional to the parasitic drag coefficient C_{dp} and missile cross-sectional area S_{REF} and is given by [13]

$$F_{dp} = QC_{dp}S_{REF}. \tag{4.7}$$

Q is defined as dynamic pressure, which depends on missile speed v_M and altitude. Dynamic pressure Q changes with the density of the air ρ and missile speed v_M and is [15]

$$Q = \frac{\rho \|v_M\|^2}{2}. \tag{4.8}$$

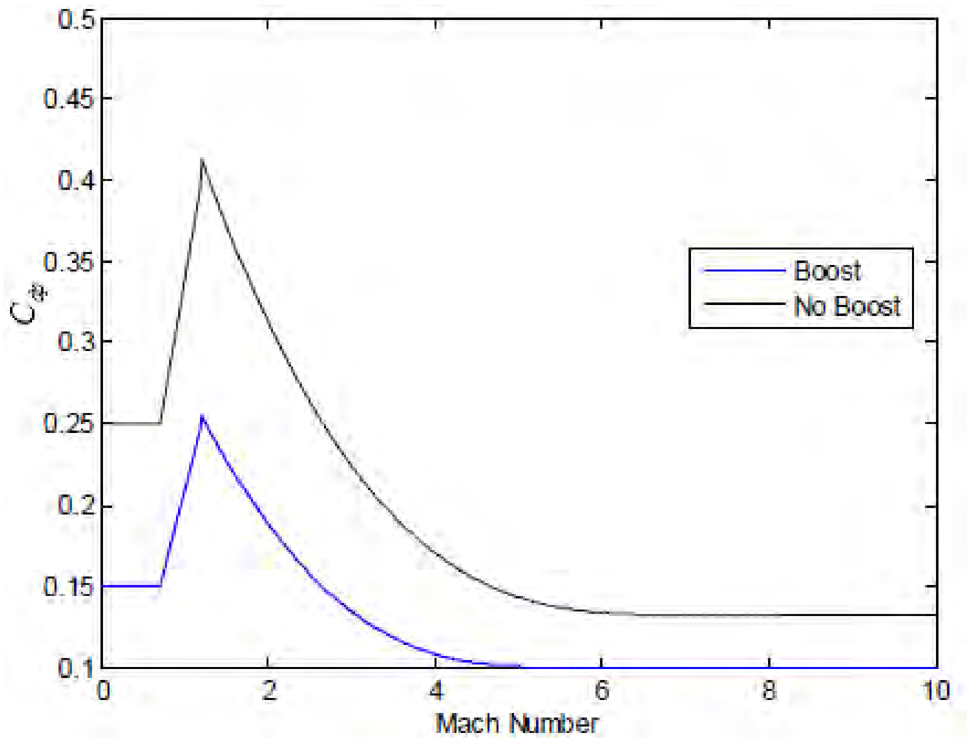


Figure 6. Parasitic drag coefficient change depending on Mach number, from [14].

b. Induced Drag

Induced drag is the drag related to the generation of lift and accounts for friction caused by the commanded guidance acceleration. Typically, induced drag is calculated using the angle-of-attack (AOA) between the missile and airflow. In this study assumptions were made based on the drag coefficient profile of the modeled Patriot missile shown in Figure 7. The drag coefficients presented in [25] are obtained by Digital DATCOM aerodynamic design software. As the angle-of-attack increases, the drag coefficient increases four to five times relative to the drag coefficient when AOA is zero.

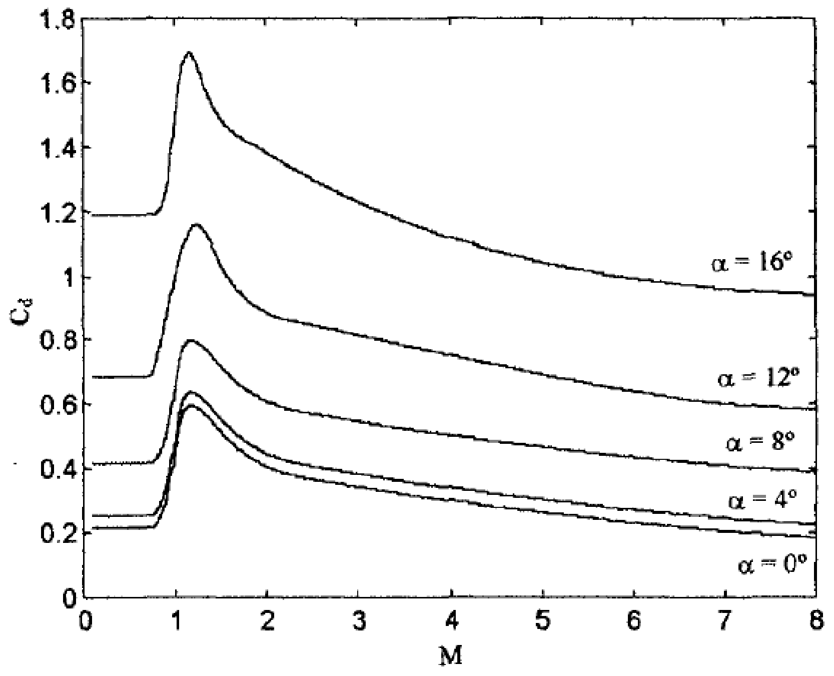


Figure 7. Drag coefficient of PAC-3 increases up to four to five times depending on angle of attack, from [25].

Induced drag is generated up to four times the parasitic drag force depending on the ratio of the commanded guidance acceleration magnitude $\|a_c\|$ to the maximum lateral acceleration capability and is given by

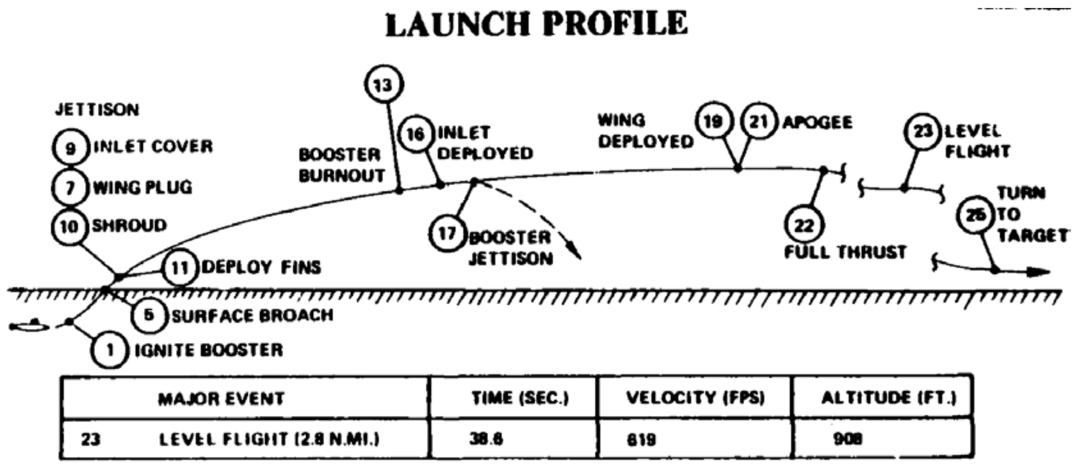
$$F_{di} = 4F_{dp}\frac{\|a_c\|}{\max g}. \tag{4.9}$$

c. Thrust and Drag Model Validation

The validation of the thrust and drag model is obtained by checking the maximum speed of the PAC-3 missile, which should be 1700 m/s [6]. Maximum speed was in the range of 1650 and 1725 m/s for all simulations.

B. TARGET MODEL

In the simulations, Tomahawk cruise missile characteristics were used to design the target model. Assumptions for the target motion and turn are described in this section. The target is modeled as a simple point mass in the NED coordinate system which has the origin at the missile's launch point. The target's initial location is set on the positive *x*-axis far from the missile depending on the range being tested. For all 3DOF simulations, the target maintained a constant speed at constant altitude. The target speed was 249.6312 m/s (819 fps) and altitude was 276.7584 meters (908 feet), as shown in Figure 8 [26]. The target maintained a straight-line trajectory until three seconds before impact. The effects of drag and gravity were not modeled for the target.



MAJOR EVENT	TIME (SEC.)	VELOCITY (FPS)	ALTITUDE (FT.)
23 LEVEL FLIGHT (2.8 N.MI.)	38.6	819	908

Figure 8. Launch profile of Tomahawk Cruise Missile, from [26].

At three seconds prior to impact, the target initiated a 6.89-g turn, which is the maximum lateral acceleration for a Turbofan engine–powered Tomahawk [26].

1. Target Motion

Using the same concept as in missile motion, we get the target state vector $T(k)$ with position and velocity components along the *x*, *y*, and *z* axes at any time step k as

$$T(k)=\begin{bmatrix} x_T(k) \\ \dot{x}_T(k) \\ y_T(k) \\ \dot{y}_T(k) \\ z_T(k) \\ \dot{z}_T(k) \end{bmatrix}. \tag{4.10}$$

When there is no turn initiated for the target, straight line target motion was generated using the target transition matrix F_T and the target state vector at next time step $T(k+1)$ specified by

$$T(k+1)=F_T T(k)=\begin{bmatrix} 1 & \Delta & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \Delta & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x_T(k) \\ \dot{x}_T(k) \\ y_T(k) \\ \dot{y}_T(k) \\ z_T(k) \\ \dot{z}_T(k) \end{bmatrix}. \tag{4.11}$$

2. Target Turn

Time-to-go t_{go} is defined as the time before the impact and is calculated by dividing range by the closing speed. For PN guidance when $N'=5$, when the target starts maneuvering two to three seconds before the intercept gives the highest miss distance [15]. To follow and improve on previous studies, target turn start time was set to three seconds, as in Broadston and Osborn [12], [14]. A target turn of $\|a_T\|$=6.89 g is implemented on the target when t_{go} is computed as three seconds.

In reality, when an aircraft starts a turn, it reaches the maximum turn acceleration after a time, and turning at the same altitude causes it to decrease its speed. In this study, delay in acceleration and decrease in speed is neglected; hence, target turn is modeled as a step maneuver. During the turn, the target maintains its flight altitude and its speed. The target keeps turning until the simulation stops.

The target can increase the magnitude of line-of-sight rate $\left\|\dot{\theta}_L\right\|$ the most by turning its velocity vector reversely proportional to the LOS rate [11]; hence, the turn of the target is presented as into the missile's trajectory. The target maneuver is generated by modifying the target transition matrix into F_{Turn} to form [19]

$$T(k+1) = F_{Turn} T(k)$$
$$= \begin{bmatrix} 1 & \sin(\omega_{turn}\Delta)/\omega_{turn} & 0 & (1-\cos(\omega_{turn}\Delta))/\omega_{turn} & 0 & 0 \\ 0 & \cos(\omega_{turn}\Delta) & 0 & -\sin(\omega_{turn}\Delta) & 0 & 0 \\ 0 & (1-\cos(\omega_{turn}\Delta))/\omega_{turn} & 1 & \sin(\omega_{turn}\Delta)/\omega_{turn} & 0 & 0 \\ 0 & \sin(\omega_{turn}\Delta) & 0 & \cos(\omega_{turn}\Delta) & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_T(k) \\ \dot{x}_T(k) \\ y_T(k) \\ \dot{y}_T(k) \\ z_T(k) \\ \dot{z}_T(k) \end{bmatrix}. \quad (4.12)$$

C. NOISE MODEL

The simulated missile in our model is equipped with onboard active radar, which can measure noisy range and LOS angles. The noise model adds sensor errors to the actual measurements. The noise used in the simulations discussed in this thesis is represented in the same manner as was done in previous work by Osborn [14].

Noise is added to the actual range, LOS azimuth angle, and LOS elevation angle measurements by using the randn.m function in MATLAB. The noisy LOS azimuth, LOS elevation angle, and range measurements in the 3DOF model are generated, respectively, with

$$\theta^*_{L_{az}} = \theta_{L_{az}} + \sigma_{\theta_L} n_{rand}, \quad (4.13)$$

and

$$\theta^*_{L_{el}} = \theta_{L_{el}} + \sigma_{\theta_L} n_{rand}, \quad (4.14)$$

and

$$r^* = r + \sigma_r n_{rand} \quad (4.15)$$

where upper script * denotes a noisy measurement, and σ_{θ_L} and σ_r are the standard deviations of the sensor measurements of LOS angles and range, respectively. The parameter n_{rand} is a pseudorandom number obtained from a Gaussian random number generator in MATLAB with variance of one. Variance for the measurements is increased by multiplying the baseline standard deviations by a noise factor, as given by

$$\sigma_{\theta_{Laz}} = f_{noise}\sigma_{\theta_{L(base)}}, \tag{4.16}$$

and

$$\sigma_{\theta_{Lel}} = f_{noise}\sigma_{\theta_{L(base)}}, \tag{4.17}$$

and

$$\sigma_r = f_{noise}\sigma_{r(base)} \tag{4.18}$$

where f_{noise} is a multiplying factor used to systematically increase noise for the measurements. This factor is mainly used to examine the maximum level of noise that a guidance law can tolerate and still intercept the target 70% of the time.

For the 3DOF model, the baseline simulated sensor standard deviation for range $\sigma_{r(base)}$ is defined as 10.0 meters; simulated sensor standard deviation for LOS azimuth and elevation angles $\sigma_{\theta_{L(base)}}$ is defined as 1.0 mrad. These assumptions were also made by Pehr [13] and Osborn [14].

D. FILTER IMPLEMENTATION

The Kalman filtering technique is used to obtain the necessary parameters to implement PN and DG guidance laws. In the filtered-noise study section, only PN and DG are examined. APN was discarded from comparison because, from the noiseless environment preliminary results, APN showed unsatisfactory behavior for all ranges tested. The four filters and the modifications used in the 3DOF model are described in this section. These filters are adapted from Osborn’s study [14]. They are divided into range and bearing filters, which estimate time derivatives of LOS angles and range based on noisy measurements received from the simulated missile’s onboard sensors.

Modifications need to be made to the three vectors to get filtered measurements out of the noisy sensor measurements. These vectors—the state vector, process noise covariance vector, and deterministic input vector—are implemented differently for each guidance law. For PN, two-dimensional range and bearing filters are used. Two-dimensional filters estimate the range and bearing and their first time derivative. For DG, additional target parameters are needed; therefore, to estimate target acceleration $\left\|\hat{a}_T\right\|$ and target lead angle $\hat{\eta}_T$, three-dimensional range and bearing filters are used. Three-dimensional filters estimate the range and bearing and their first and second time derivatives. The outputs of the three-dimensional range and bearing filters are LOS azimuth angle acceleration $\hat{\ddot{\theta}}_{L_{az}}$, LOS elevation angle acceleration $\hat{\ddot{\theta}}_{L_{el}}$ and range acceleration $\hat{\ddot{r}}$. The simulation uses the same filter for both azimuth and elevation angle state estimates.

The characteristics of the filters, which are the state estimate, state-estimate covariance, process covariance, and deterministic inputs, are described in the following section. The differences between the two- and three-dimensional filters are also explained.

1. State Estimate

The state estimate is obtained by using the Kalman filter algorithm described in Chapter III. It is assumed that before launch the missile acquires target range and bearing measurements from the ground radar, which can provide more accurate tracking parameters than the missile's onboard seeker. For the prediction phase, the first four actual values of the parameters are used to generate the initial state estimates of range and LOS angles.

The PN guidance law only requires the missile lead angle η_M, LOS angle rate $\dot{\theta}_L$ and closing speed v_c. In the simulations, actual missile position, velocity, and acceleration are assumed to be known because PAC-3 uses an Inertial Navigation System like many tactical missiles. The estimate of η_M is computed using the missile's state vector and the estimated LOS vector, which depends on the estimated range and bearing angles. The LOS angle rate $\dot{\theta}_L$ and closing speed v_c, which is negative of range rate estimate $\hat{\dot{r}}\left(k\right)$, are estimated using two-dimensional LOS angle and range states

$$\hat{x}_{\theta_L}(k) = \begin{bmatrix} \hat{\theta}_L(k) \\ \hat{\dot{\theta}}_L(k) \end{bmatrix} \tag{4.19}$$

and

$$\hat{x}_r(k) = \begin{bmatrix} \hat{r}(k) \\ \hat{\dot{r}}(k) \end{bmatrix}. \tag{4.20}$$

Besides missile lead angle η_M, LOS angle rate $\dot{\theta}_L$, and closing speed v_c, DG guidance law requires magnitude of target acceleration $\|a_T\|$ and target lead angle η_T, which is defined as the angle between the target's velocity vector and the LOS. To provide these parameters, filters use three-dimensional state vectors

$$\hat{x}_{\theta_L}(k) = \begin{bmatrix} \hat{\theta}_L(k) \\ \hat{\dot{\theta}}_L(k) \\ \hat{\ddot{\theta}}_L(k) \end{bmatrix} \tag{4.21}$$

and

$$\hat{x}_r(k) = \begin{bmatrix} \hat{r}(k) \\ \hat{\dot{r}}(k) \\ \hat{\ddot{r}}(k) \end{bmatrix}. \tag{4.22}$$

The process to find the target acceleration $\|a_T\|$ and target lead angle η_T, which are the required parameters for DG guidance law implementation, is provided in the next section.

2. State Estimate Covariance

The state estimate covariance matrix $P(k)$ is a measure of the accuracy of the state estimate at time k. The initial value for the state covariance $P(0)$ is computed using measurement covariance matrix $R(k)$, which defines the accuracy of the simulated missile's onboard sensors. The simulation uses the same filters for LOS azimuth and elevation angles. The standard deviations of both angle measurements are the same, 1.0 mrad; therefore, their initial state-estimate covariance matrices are the same. Initial two-dimensional range and LOS angles filter covariance matrices for PN guidance law are [21]

$$P_{\theta_L}(0) = \sigma_{\theta_L}^2 \begin{bmatrix} 1 & 0 \\ 0 & \frac{2}{\Delta^2} \end{bmatrix} \tag{4.23}$$

and

$$P_r(0) = \sigma_r^2 \begin{bmatrix} 1 & 0 \\ 0 & \frac{2}{\Delta^2} \end{bmatrix}. \tag{4.24}$$

Initial three-dimensional DG filter covariance matrices are [21]

$$P_{\theta_L}(0) = \sigma_{\theta_L}^2 \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{2}{\Delta^2} & 0 \\ 0 & 0 & \frac{4}{\Delta^4} \end{bmatrix} \tag{4.25}$$

and

$$P_r(0) = \sigma_r^2 \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{2}{\Delta^2} & 0 \\ 0 & 0 & \frac{4}{\Delta^4} \end{bmatrix}. \tag{4.26}$$

3. Process Covariance

Process covariance is a sequence of zero-mean white Gaussian noise with covariance matrices $Q(k)$. Process covariance for two-dimensional PN filters is [21]

$$Q(k) = q^2(k) \begin{bmatrix} \frac{\Delta^3}{3} & \frac{\Delta^2}{2} \\ \frac{\Delta^3}{2} & \Delta \end{bmatrix}. \tag{4.27}$$

Process covariance for three-dimensional filters is [21]

$$Q(k) = q^2(k)\begin{bmatrix} \frac{\Delta^5}{20} & \frac{\Delta^4}{8} & \frac{\Delta^3}{6} \\ \frac{\Delta^4}{8} & \frac{\Delta^3}{3} & \frac{\Delta^2}{2} \\ \frac{\Delta^3}{6} & \frac{\Delta^2}{2} & \Delta \end{bmatrix}. \tag{4.28}$$

In the two matrices above, $q^2(k)$ is defined as the process covariance weighting factor. The weighting factor is changed to follow the target through any possible maneuver so that $q^2(k)$ can help to improve the filter's performance. Weighting factors are defined separately for LOS angles and range filters.

a. LOS Angle Weighting Factor

For both LOS azimuth and elevation angles, the same weighting factor is used. Covariance of the LOS angle determines the LOS angle weighting factor $q_{\theta_L}{}^2(k)$. The change in LOS angles between two time steps is [19]

$$\hat{\theta}_L(k+1) - \hat{\theta}_L(k) = \tan^{-1}\left(\frac{\left\|\hat{a}_{Tperp}(k)\right\|\Delta^2}{2\hat{r}(k)}\right) \approx \frac{\left\|\hat{a}_{Tperp}(k)\right\|\Delta^2}{2\hat{r}(k)} \tag{4.29}$$

where θ_L represents both azimuth and elevations angles and $\left\|\hat{a}_{Tperp}\right\|$ is the estimated magnitude of target acceleration perpendicular to the LOS. The process noise accounts for the hardest target maneuver acceleration; therefore, target turn acceleration magnitude perpendicular to the LOS can be specified as $\left\|\hat{a}_{Tperp}\right\| = 6.89\ g$, which is the simulated target's maximum turn acceleration. We consider the change in LOS angle, given in (4.29), to be the standard deviation of the random error in $Q(k)$. Since the change in LOS angle is very small in one time step, it is approximated that the arctangent of the value is equal to that value. Taking the square of this standard deviation gives the covariance, and making it equal to the covariance from the $Q(k)$ matrix of two-dimensional filters defined in (4.27), we obtain [14],[19]

$$\left(\hat{\dot{\theta}}(k+1)-\hat{\dot{\theta}}(k)\right)^2 = q_{\theta_L}{}^2(k)\frac{\Delta^3}{3} = \frac{\left\|\hat{a}_{Tperp}(k)\right\|^2 \Delta^4}{4\hat{r}(k)^2} \quad . \tag{4.30}$$

Solving for the LOS angle weighting factor yields [19]

$$q_{\theta_L}{}^2(k) = \frac{3\left\|\hat{a}_{Tperp}(k)\right\|^2 \Delta}{4\hat{r}(k)^2}. \tag{4.31}$$

The LOS angle weighting factor is derived from $\Delta^3/3$ term in the LOS angle covariance of the two-dimensional PN filter. The DG LOS angle filters use the same LOS angle weighting factor. As seen in (4.31), the LOS angle weighting factor changes with range. As the range gets smaller, target accelerations cause the LOS angle change between two time steps to increase. The LOS angle weighting factor $q_{\theta_L}{}^2(k)$ increases with the range estimate term in the denominator to compensate for this larger change in LOS angle due to the target turn.

DG is more sensitive to noise due to the additional acceleration terms in the guidance law; therefore, for DG filters the LOS angle weighting factor is increased by ten times in the last 500 meters. Increasing process noise covariance in the endgame, when the target maneuvers are expected, helps the filter get better estimates.

b. Range Weighting Factor

Covariance of the range for PN filters described in (4.27) determines the range weighting factor $q_r{}^2(k)$ for both two and three-dimensional Kalman filters. The change in range r between two time steps is [19]

$$\hat{r}(k+1)-\hat{r}(k) = \frac{\left\|\hat{a}_{Tpara}(k)\right\|\Delta^2}{2}, \tag{4.32}$$

where $\left\|\hat{a}_{Tpara}\right\|$ is the estimated magnitude of target acceleration parallel to the LOS. To make the process noise account for the hardest target maneuver acceleration, target turn acceleration magnitude parallel to the LOS can be specified as $\left\|\hat{a}_{Tpara}\right\| = 6.89g$, which is the simulated target's maximum turn acceleration. As seen in (4.32), change in range is independent from range, which means that the range weighting factor is constant at all times. We consider the change in range, which is given in (4.32), to be the standard deviation of the random error in $Q(k)$ for the range filters. Taking the square of this standard deviation gives the covariance, and equating it to the covariance of range from the $Q(k)$ matrix described in (4.27), we obtain [14], [19]

$$\left(\hat{r}(k+1)-\hat{r}(k)\right)^2 = q_r^{\,2}(k)\frac{\Delta^3}{3} = \frac{\left\|\hat{a}_{Tpara}(k)\right\|^2 \Delta^4}{4}. \tag{4.33}$$

Solving for the range weighting factor yields [19]

$$q_r^{\,2}(k) = \frac{3\left\|\hat{a}_{Tpara}(k)\right\|^2 \Delta}{4}. \tag{4.34}$$

4. Deterministic Inputs

We can assume that missile motion, which is caused by missile thrust and drag accelerations, affect the changes of range and LOS angles. These changes are implemented as deterministic input vector $u(k)$ to the filter estimate to improve the filter's performance. Changes in the position, velocity, and acceleration of the state are calculated and added in the predicted state estimate equation (3.7).

Adding missile thrust acceleration $a_{Mthrust}$ and drag a_{Mdrag} together gives the deterministic acceleration a_u. It should be noted that the directions of missile thrust acceleration vector and missile drag acceleration vector are opposite to each other but parallel to the missile velocity vector. The deterministic acceleration is [19]

$$a_u = a_{Mthrust} + a_{Mdrag}. \tag{4.35}$$

The portion of the deterministic acceleration parallel to the LOS affects the range state estimation. The magnitude of the first two components of the perpendicular portion of deterministic acceleration vector is used for the LOS azimuth state estimation. The last component of the perpendicular portion of the deterministic acceleration affects the LOS elevation angle state estimation.

The first two components of the deterministic acceleration vector for both PN and DG filters are the same. For DG filters, modifications are done to get the estimated range acceleration and LOS angle accelerations.

a. LOS Angle Deterministic Input

The portion of the deterministic acceleration perpendicular to the LOS affects the LOS angles state estimation and is denoted as $a_{u_{perp}}$. For the LOS azimuth state estimation, the first two components of the perpendicular portion of deterministic acceleration is used. The last component of the perpendicular portion of the deterministic acceleration affects the LOS elevation angle state estimation. The change in LOS angle θ_L is found in the same manner described in (4.29), and the first time derivative of (4.29) gives the change in the LOS angle rate $\dot{\theta}_L$, which is specified as [19]

$$\hat{\dot{\theta}}_L(k+1)-\hat{\dot{\theta}}_L(k)=\frac{4a_{u_{perp}}(k)\hat{r}(k)\Delta}{\left(4\hat{r}(k)^2+a_{u_{perp}}(k)^2\Delta^4\right)}. \tag{4.36}$$

Putting (4.29) and (4.36) together gives the estimated two-dimensional LOS angle deterministic input [14]

$$\hat{u}_{\theta_L}(k)=\begin{bmatrix}\tan^{-1}\left(\dfrac{a_{u_{perp}}(k)\Delta^2}{2\hat{r}(k)}\right)\\[2ex] \dfrac{4a_{u_{perp}}(k)\hat{r}(k)\Delta}{\left(4\hat{r}(k)^2+a_{u_{perp}}(k)^2\Delta^4\right)}\end{bmatrix}. \tag{4.37}$$

For three-dimensional DG LOS angle filters, the deterministic input is [19]

$$\hat{u}_{\theta_L}(k)=\begin{bmatrix}\tan^{-1}\left(\dfrac{a_{u_{perp}}(k)\Delta^2}{2\hat{r}(k)}\right)\\[2ex] \dfrac{4a_{u_{perp}}(k)\hat{r}(k)\Delta}{\left(4\hat{r}(k)^2+a_{u_{perp}}(k)^2\Delta^4\right)}\\[2ex] 0\end{bmatrix}. \tag{4.38}$$

The third component of the deterministic input is zero because the acceleration portion of the LOS angle deterministic input vector is not added to the predicted state estimate in the filter; instead it is added to the corrected state estimate equation (3.10), which yields [14]

$$\hat{x}\left(k+1\middle|k+1\right)=\hat{x}\left(k+1\middle|k\right)+W\left(k+1\right)\tilde{z}\left(k+1\right)+\begin{bmatrix}0\\0\\a_{u_{perp}}\left(k\right)/\hat{r}(k)\end{bmatrix}. \tag{4.39}$$

b. Range Deterministic Input

The portion of the deterministic input parallel to the LOS affects the range state estimation and is denoted as $a_{u_{para}}$. The change in range is found in the same manner described in (4.32), and combining it with its first time derivative, which is the change in range rate $\dot{r}$, gives the two-dimensional range filter deterministic input as [19]

$$u_r(k)=a_{u_{para}}\left(k\right)\begin{bmatrix}\dfrac{\Delta^2}{2}\\\Delta\end{bmatrix}. \tag{4.40}$$

For three-dimensional LOS angle filter, the deterministic input is [19]

$$u_r(k)=a_{u_{para}}\left(k\right)\begin{bmatrix}\dfrac{\Delta^2}{2}\\\Delta\\0\end{bmatrix}. \tag{4.41}$$

The third component of the deterministic input for the range filter is also set to zero, as is done in the LOS angle deterministic input [14]. The acceleration portion of the range deterministic input vector is added to the corrected state estimate equation (3.10), which yields

$$\hat{x}\left(k+1\middle|k+1\right)=\hat{x}\left(k+1\middle|k\right)+W\left(k+1\right)\nu\left(k+1\right)+\begin{bmatrix}0\\0\\a_{u_{para}}\left(k\right)\end{bmatrix}. \tag{4.42}$$

E. GUIDANCE LAW IMPLEMENTATION

The steps used to find the required parameters for three-dimensional guidance law implementation are provided in this section. The parameters are the guidance command vector, an estimate of LOS angle rate, target and missile lead angles, and an estimate of target acceleration. Previous studies [12]–[14] applied guidance laws in the *x*-*y* plane. In this study, the SAM missile is launched from ground level and changes altitude in flight, so the commanded guidance acceleration vector needs to be applied using three-dimensional geometry.

First, the steps to derive the required parameters for guidance laws in a noiseless environment are provided. When using the actual measurements, the vectors in the guidance law implementation are defined as follows.

LOS vector, which joins missile and target positions, is

$$R(k) = \begin{bmatrix} x_T(k) \\ y_T(k) \\ z_T(k) \end{bmatrix} - \begin{bmatrix} x_M(k) \\ y_M(k) \\ z_M(k) \end{bmatrix}. \tag{4.43}$$

The LOS velocity vector is

$$\dot{R}(k) = V_r(k) = V_T(k) - V_M(k), \tag{4.44}$$

where missile velocity vector is

$$V_M(k) = \begin{bmatrix} \dot{x}_M(k) \\ \dot{y}_M(k) \\ \dot{z}_M(k) \end{bmatrix} \tag{4.45}$$

and target velocity vector is

$$V_T(k) = \begin{bmatrix} \dot{x}_T(k) \\ \dot{y}_T(k) \\ \dot{z}_T(k) \end{bmatrix}. \tag{4.46}$$

The range from the missile to the target is $r(k) = \left\| R(k) \right\|$, so the unit vector in the direction of LOS vector is

$$n_R(k) = \frac{R(k)}{\left\| R(k) \right\|}. \tag{4.47}$$

Also, the unit vector in the direction of missile velocity is

$$n_{V_M}(k) = \frac{V_M(k)}{\left\| V_M(k) \right\|}. \tag{4.48}$$

The unit vector in the direction of target velocity is

$$n_{V_T}(k) = \frac{V_T(k)}{\left\| V_T(k) \right\|}. \tag{4.49}$$

The closing velocity vector between the missile and the target becomes

$$V_c(k) = -\left(V_r(k) \bullet n_R(k)\right) n_R(k). \tag{4.50}$$

The closing speed is defined as

$$v_c(k) = -\dot{r}(k) = -\left(V_r(k) \bullet n_R(k)\right). \tag{4.51}$$

The perpendicular portion of LOS velocity vector $V_r(k)$ to LOS vector $R(k)$ is given by [19]

$$V_p(k) = V_r(k) + V_c(k) \tag{4.52}$$

and its unit vector is

$$n_{V_p}(k) = \frac{V_p(k)}{\left\| V_p(k) \right\|}. \tag{4.53}$$

The perpendicular component of LOS velocity vector $V_p(k)$ indicates how $R(k)$ rotates about the origin in the plane defined by two vectors $R(k)$ and $V_r(k)$. If the magnitude of $V_p(k)$ is zero, $R(k)$ reaches the target without requiring any guidance acceleration command [19]; hence, we can define the magnitude of the LOS rate as

$$\left|\dot{\theta}_L(k)\right| = \frac{\left\| V_p(k) \right\|}{r(k)}. \tag{4.54}$$

As discussed in Chapter II, true PN guidance can be specified as

$$a_c(k) = N'V_c(k)\left|\dot{\theta}_L(k)\right| n_{V_p}(k) \tag{4.54}$$

where $a_c(k)$ is the guidance acceleration vector, which is applied perpendicular to LOS vector $R(k)$. As in pure PN, guidance command should be applied perpendicular to the missile velocity vector for practical purposes; hence, the guidance law becomes

$$a_c(k) = \frac{N' V_c(k) \left|\dot{\theta}_L(k)\right|}{\cos(\eta_M)} n_c(k) \tag{4.56}$$

where η_M is missile lead angle and is defined as

$$\eta_M = \cos^{-1}(n_R \bullet n_{V_M}). \tag{4.57}$$

Pure PN uses $n_c(k)$ as the guidance acceleration command vector, which is the unit vector that lies in the $V_p(k)$ and $V_M(k)$ plane and is perpendicular to $V_M(k)$. Pure PN guidance acceleration command vector $n_c(k)$ is defined as

$$n_c(k) = n_{V_M}(k) \times \left(n_{V_p}(k) \times n_{V_M}(k) \right). \tag{4.58}$$

Since other guidance laws examined in this study are all augmentations of the PN guidance law, guidance command is applied perpendicular to the missile velocity vector, in the direction of $n_c(k)$, and the magnitude of the guidance command is increased by the $\cos(\eta_M)$ term in the denominator.

The implementation of APN and DG guidance requires the magnitude of the target's acceleration $\|a_T\|$. Additionally, the DG guidance law requires the target lead angle η_T. In a noiseless environment, an estimate of the target acceleration can be extrapolated easily from

$$\left\|\hat{a}_T(k)\right\| = \frac{V_T(k) - V_T(k-1)}{\Delta}, \tag{4.59}$$

and target lead angle η_T is computed as

$$\eta_T = \cos^{-1}(n_R \bullet n_{V_T}). \tag{4.60}$$

When the filter is applied to noisy measurements, the guidance command vector and estimate of LOS angle rate are extrapolated as follows.

The estimate of the LOS vector and its unit vector are generated from the filtered range, LOS azimuth angle, and LOS elevation angle. The estimate of the LOS vector is

$$\widehat{R}(k) = \begin{bmatrix} \hat{r}(k)\cos(\hat{\theta}_{L_{el}})\cos(\hat{\theta}_{L_{az}}) \\ \hat{r}(k)\cos(\hat{\theta}_{L_{el}})\sin(\hat{\theta}_{L_{az}}) \\ \hat{r}(k)\sin(\hat{\theta}_{L_{el}}) \end{bmatrix} \tag{4.61}$$

and its unit vector is

$$\hat{n}_R(k) = \frac{\widehat{R}(k)}{\hat{r}(k)}. \tag{4.62}$$

Missile velocity is already known, and adding the missile velocity vector to the estimated LOS rate vector gives the estimate of the target velocity

$$\widehat{V}_T\left(k\right) = V_M\left(k\right) + \hat{\dot{R}}(k), \tag{4.63}$$

where estimated LOS rate vector $\hat{\dot{R}}(k)$, which is also defined as estimated LOS velocity vector $\widehat{V_r}(k)$, is

$$\widehat{V_r}(k) = \hat{\dot{R}}(k) = \frac{R\left(k\right) - R\left(k-1\right)}{\Delta} =$$
$$\begin{bmatrix} \cos(\hat{\theta}_{L_{az}})\cos(\hat{\theta}_{L_{el}})\hat{\dot{r}} - \cos(\hat{\theta}_{L_{el}})\sin(\hat{\theta}_{L_{az}})\hat{r}\hat{\dot{\theta}}_{L_{az}} - \cos(\hat{\theta}_{L_{az}})\sin(\hat{\theta}_{L_{el}})\hat{r}\hat{\dot{\theta}}_{L_{el}} \\ \cos(\hat{\theta}_{L_{el}})\sin(\hat{\theta}_{L_{az}})\hat{\dot{r}} + \cos(\hat{\theta}_{L_{az}})\cos(\hat{\theta}_{L_{el}})\hat{r}\hat{\dot{\theta}}_{L_{az}} - \sin(\hat{\theta}_{L_{az}})\sin(\hat{\theta}_{L_{el}})\hat{r}\hat{\dot{\theta}}_{L_{el}} \\ \sin(\hat{\theta}_{L_{el}})\hat{\dot{r}} + \cos(\hat{\theta}_{L_{el}})\hat{r}\hat{\dot{\theta}}_{L_{el}} \end{bmatrix}. \tag{4.64}$$

The same steps as in the noiseless environment followed to find $\widehat{V_p}(k)$, which is the portion of estimated LOS velocity vector $\widehat{V_r}(k)$ perpendicular to the estimated LOS vector $\widehat{R}(k)$ and is given by

$$\widehat{V_p}(k) = \widehat{V_r}(k) + \widehat{V_c}(k) = \hat{\dot{R}}(k) + \hat{\dot{r}}(k) \bullet \hat{n}_R(k) \tag{4.65}$$

where its unit vector is

$$\hat{n}_{V_p}(k) = \frac{\widehat{V_p}(k)}{\left\|\widehat{V_p}(k)\right\|}. \tag{4.66}$$

Estimated LOS angle rate and guidance acceleration command vectors, which are the required parameters to implement the guidance laws, can be found, respectively, from

$$\left|\widehat{\dot{\theta}_L}(k)\right| = \frac{\left\|\widehat{V_p}(k)\right\|}{\hat{r}(k)} \tag{4.67}$$

and

$$n_c(k) = n_{V_M}(k) \times \left(\widehat{n_{V_p}}(k) \times \widehat{n_R}(k)\right). \tag{4.68}$$

For the DG guidance law, we obtain estimates of the LOS azimuth angle acceleration $\hat{\ddot{\theta}}_{L_{az}}$, LOS elevation angle acceleration $\hat{\ddot{\theta}}_{L_{el}}$, and range acceleration $\hat{\ddot{r}}$ by using a three-dimensional filter. The DG guidance law requires the magnitude of the target's acceleration $\left\|\hat{a}_T(k)\right\|$ and target lead angle $\hat{\eta}_T$. These parameters are extrapolated from the three-dimensional filter outputs.

The estimate of the target acceleration magnitude, which is necessary for DG guidance law, is

$$\left\|\widehat{a_T}(k)\right\| = \left\|M_a(k) + \hat{\ddot{R}}(k)\right\|, \tag{4.69}$$

where LOS acceleration vector $\hat{\ddot{R}}(k)$ is the second time derivative of LOS vector $\hat{R}(k)$. LOS acceleration vector $\hat{\ddot{R}}(k)$ is specified as

$$\hat{\ddot{R}}(k) = \begin{bmatrix} \text{-}\sin(\hat{\theta}_{L_{az}})\left(\left(\hat{\ddot{\theta}}_{L_{az}}\hat{r} + 2\hat{\dot{\theta}}_{L_{az}}\hat{\dot{r}}\right)\cos(\hat{\theta}_{L_{el}}) \text{-} 2\hat{\dot{\theta}}_{L_{az}}\hat{\dot{\theta}}_{L_{el}}\hat{r}\sin(\hat{\theta}_{L_{el}})\right) - \ldots \\ \ldots\cos(\hat{\theta}_{L_{az}})\left(\left(\left(\hat{\dot{\theta}}_{L_{az}}^2 + \hat{\dot{\theta}}_{L_{el}}^2\right)\hat{r} \text{-} \hat{\ddot{r}}\right)\cos(\hat{\theta}_{L_{el}}) + \left(\hat{\ddot{\theta}}_{L_{el}}\hat{r} + 2\hat{\dot{\theta}}_{L_{el}}\hat{\dot{r}}\right)\sin(\hat{\theta}_{L_{el}})\right) \\ \cos(\hat{\theta}_{L_{az}})\left(\left(\hat{\ddot{\theta}}_{L_{az}}\hat{r} + 2\hat{\dot{\theta}}_{L_{az}}\hat{\dot{r}}\right)\cos(\hat{\theta}_{L_{el}}) \text{-} 2\hat{\dot{\theta}}_{L_{az}}\hat{\dot{\theta}}_{L_{el}}\hat{r}\sin(\hat{\theta}_{L_{el}})\right) - \ldots \\ \ldots\sin(\hat{\theta}_{L_{az}})\left(\left(\left(\hat{\dot{\theta}}_{L_{az}}^2 + \hat{\dot{\theta}}_{L_{el}}^2\right)\hat{r} \text{-} \hat{\ddot{r}}\right)\cos(\hat{\theta}_{L_{el}}) + \left(\hat{\ddot{\theta}}_{L_{el}}\hat{r} + 2\hat{\dot{\theta}}_{L_{el}}\hat{\dot{r}}\right)\sin(\hat{\theta}_{L_{el}})\right) \\ \left(\hat{\ddot{\theta}}_{L_{el}}\hat{r} + 2\hat{\dot{\theta}}_{L_{el}}\hat{\dot{r}}\right)\cos(\hat{\theta}_{L_{el}}) + \left(-\hat{\dot{\theta}}_{L_{el}}^2\hat{r} + \hat{\ddot{r}}\right)\sin(\hat{\theta}_{L_{el}}) \end{bmatrix}. \tag{4.70}$$

F. SUMMARY

In this chapter, we have examined the simulation methodology of the 3DOF model. The missile's motion, thrust, and drag were modeled using the PAC-3 specifications. The thrust model of the simulated SAM, which was launched from ground level, was explained. The target was simulated as a low-level flying cruise missile at a constant speed by using the Tomahawk cruise missile flight characteristics. A noise model, which represents the noisy sensor measurements, was presented. Four Kalman filters used in the simulations were described by their state estimate, state-covariance estimate, process covariance, and deterministic inputs. Finally, implementation of PN and DG guidance laws in three-dimensional geometry was explained.

THIS PAGE INTENTIONALLY LEFT BLANK

V. RESULTS AND PERFORMANCE ANALYSIS

The simulation results and performance analysis of each guidance law at three different ranges and against maneuvering and non-maneuvering targets for both noisy and noiseless environments are included in this chapter. The maximum noise factors for two of the three guidance laws are also presented; the third law does not provide adequate performance in a noisy environment.

In this study, the intercept of a CM by a SAM launched from ground level and requiring true three-dimensional guidance is simulated. It was convenient to run the simulations first with a non-maneuvering target in a noiseless environment. Then, the simulations were run against maneuvering targets to see the effect of additional target-related terms in the APN and DG guidance laws. To achieve the main objective of this thesis, simulations with Kalman filtered noise were run against maneuvering targets at 2, 8, and 15 km ranges. Additionally, the maximum noise tolerance to achieve at least a 70% hit probability is tested for PN and DG guidance laws. All the test scenarios in this study are shown in Table 2.

Table 2. Simulation scenarios include different environments at various ranges (M is maneuvering, NM is non-maneuvering).

	2km		8 km		15 km	
	M	NM	M	NM	M	NM
Noiseless Environment	X	X	X	X	X	X
With Kalman Filtered Noise	X		X		X	
Maximum Noise Factor	X		X		X	

The performance analysis of guidance laws for the different scenarios is presented differently than in previous research. Broadston, Pehr and Osborn [12]–[14] used the KB method to compare guidance laws. Broadston defined the KB as the maximum range at which a missile can reach the kill radius of the target as a function of aspect angle. Performance of guidance laws can be determined using other parameters. In this study, the analysis is based on three parameters: divert, impact time and impact velocity. *Divert* is defined as the integral of magnitude of the commanded guidance acceleration over the entire missile flight time. It is important for tactical missiles such as PAC-3 because it is a measure of fuel expenditure. *Impact time* is desired to be short in order to intercept the target as soon as possible. The third parameter used for performance analysis is the *impact velocity*, which is also related to divert. Applying less guidance acceleration to the missile exposes the missile to less drag. The missile slows down less due to drag and gravity. As a consequence, the missile has a higher impact velocity.

A. SIMULATION RESULTS FOR NOISELESS ENVIRONMENT

In a noiseless environment, all three guidance laws show the same behavior against a non-maneuvering target. The reason is that the target acceleration terms in (2.3) and (2.5) are equal to zero and have no effect on APN and DG guidance laws. The results from this simulation show that divert, impact time and impact velocity are the same for all three guidance laws.

Against maneuvering targets the simulations are run at various ranges. It is expected that APN and DG will yield better results because they have additional target related terms in their formula. The results are presented in three parts depending on the range tested.

1. Against Maneuvering Targets at 2000 Meters

It can be seen in Figure 9 and Figure 10 that APN has an unsatisfactory performance in terms of divert as compared to DG and PN against maneuvering targets, which are initially located at 2000 meters. The DG guidance law causes less divert on the missile and provides better performance against outgoing targets with up to 86 degrees of target heading.

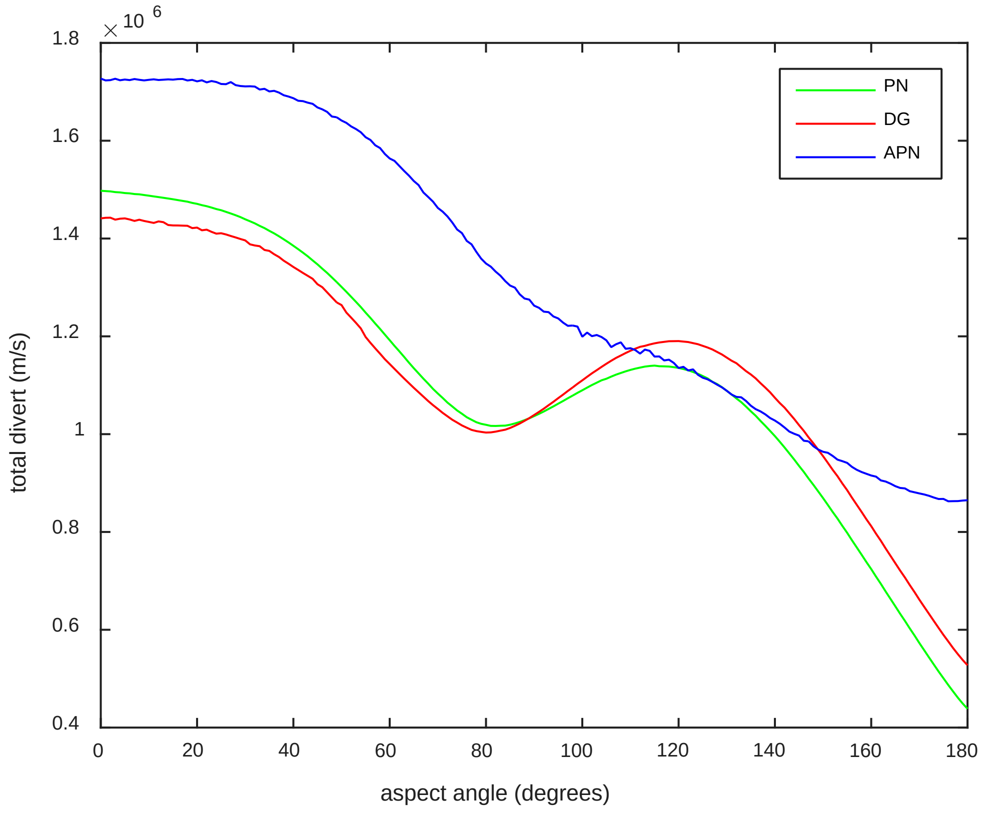


Figure 9. Comparison of divert between guidance laws against maneuvering targets at 2000 meters in a noiseless environment.

In Figure 10, it is clearly shown that DG performs better than PN and APN in tail-chase scenarios. PN works better against incoming targets if the target's heading is more than 89 degrees. It is also noticeable that APN has the same performance as PN at 126 degrees. For the incoming targets, APN performs better than DG between 111–149 degrees, as shown in Figure 10.

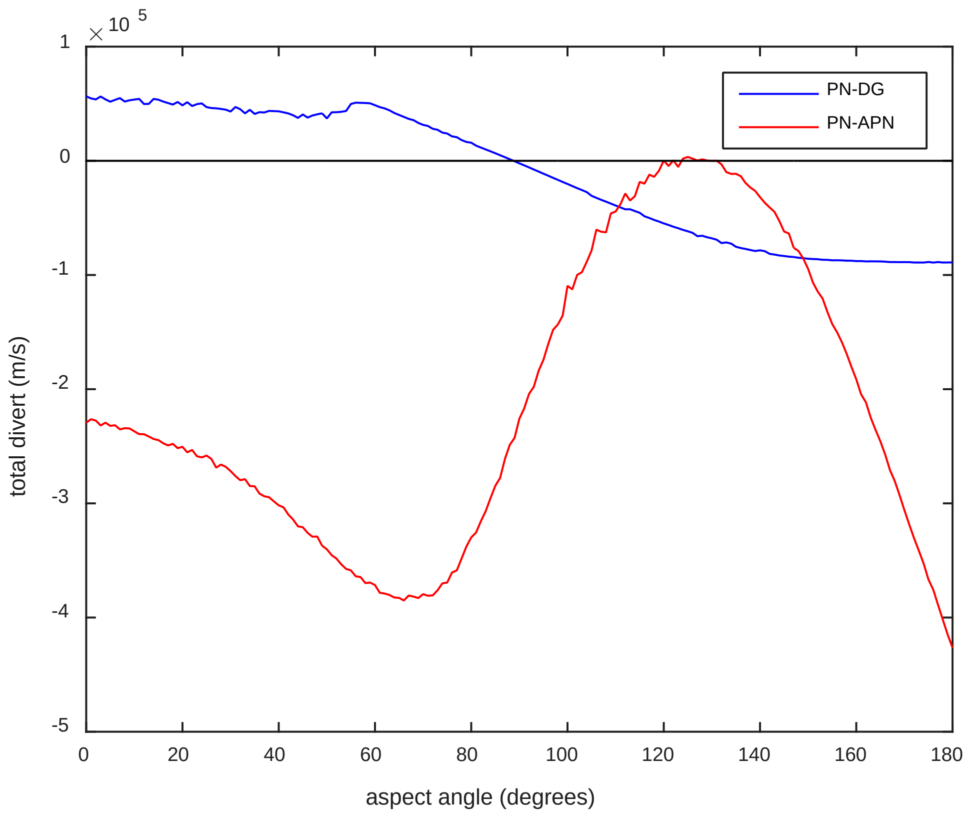


Figure 10. Difference in divert between guidance laws against maneuvering targets at 2000 meters in a noiseless environment.

It is useful to compare the impact time and the impact velocity to justify the results from the divert comparison. It is clearly seen in Figure 11 that the impact time difference between PN and DG is not significant; however, DG has a shorter impact time in the aft quadrant, which agrees with the divert comparison result. The impact time difference between PN and APN is very small compared to the overall impact time (0.025 << 3.75–6.75 seconds).

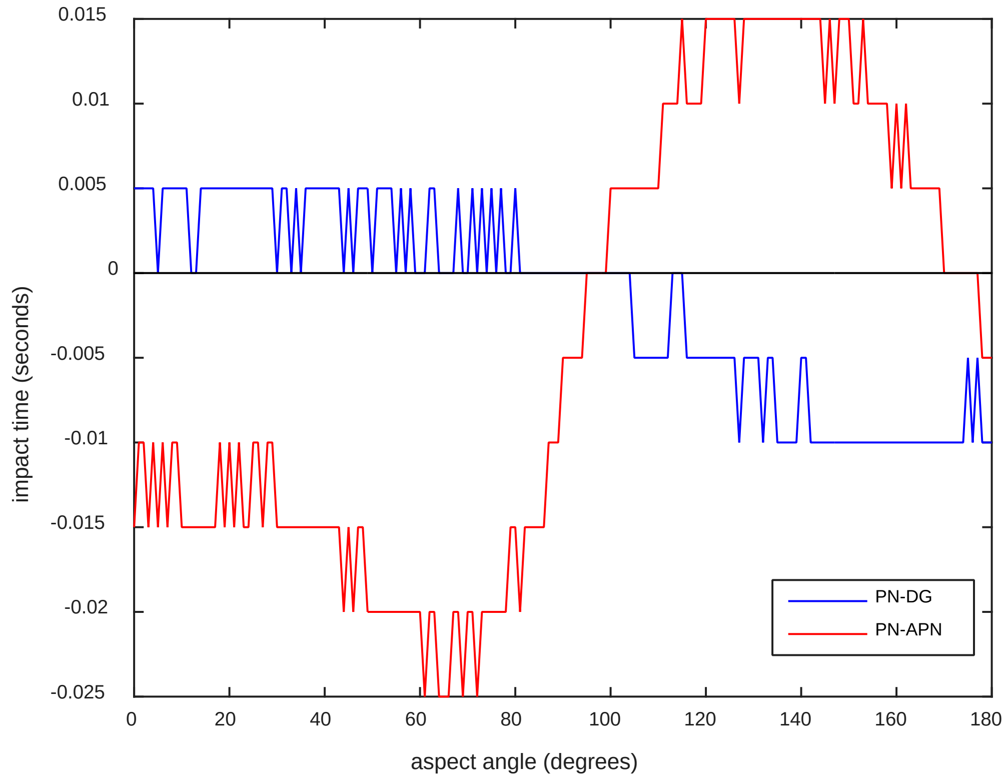


Figure 11. Difference in impact time between guidance laws against maneuvering targets at 2000 meters in a noiseless environment.

As described in the first part of this chapter, if the guidance law requires less divert, the missile generally has a higher impact velocity. Making the comparison due to the third performance parameter, impact velocity, we obtain similar results as found for the divert comparison. DG gives a higher impact velocity up to 89 degrees, as shown clearly in Figure 12. A missile using APN has lower impact velocity than a missile using PN guidance against maneuvering targets coming from all aspect angles.

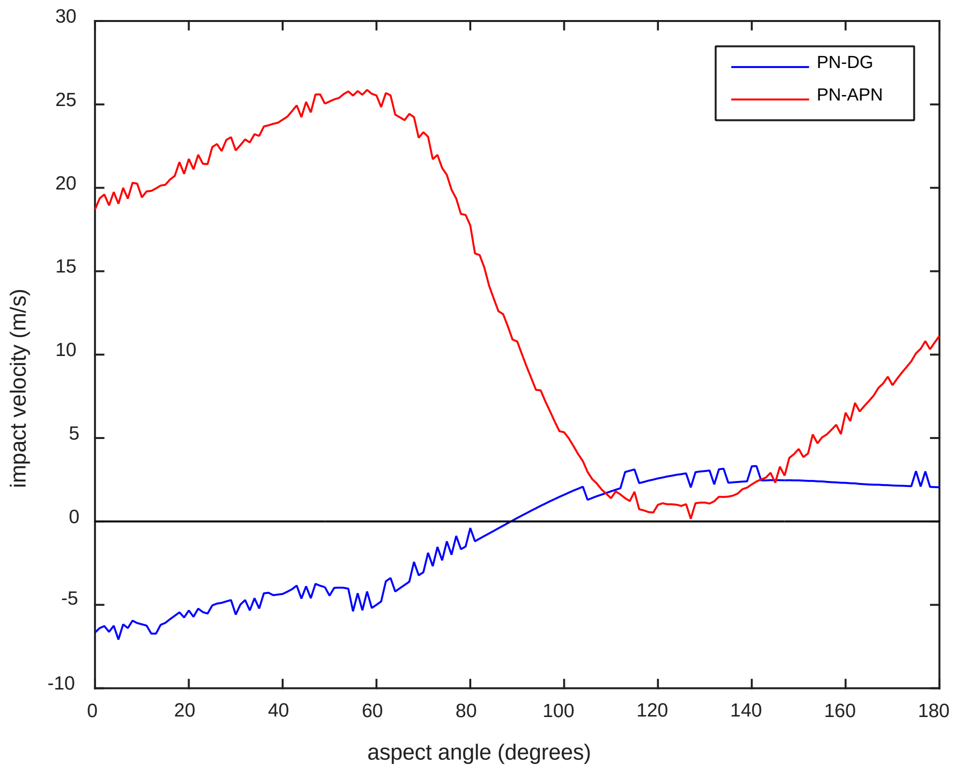


Figure 12. Difference in impact time between guidance laws against maneuvering targets at 2000 meters in a noiseless environment.

2. Against Maneuvering Targets at 8000 Meters

For 8000 meters, the results are consistent with the results from previous test at 2000 meters. DG performs better in the aft quadrant up to 76 degrees, using divert as the performance analysis parameter. Performance of APN is inferior and cannot compete with APN and DG, as shown in Figure 13.

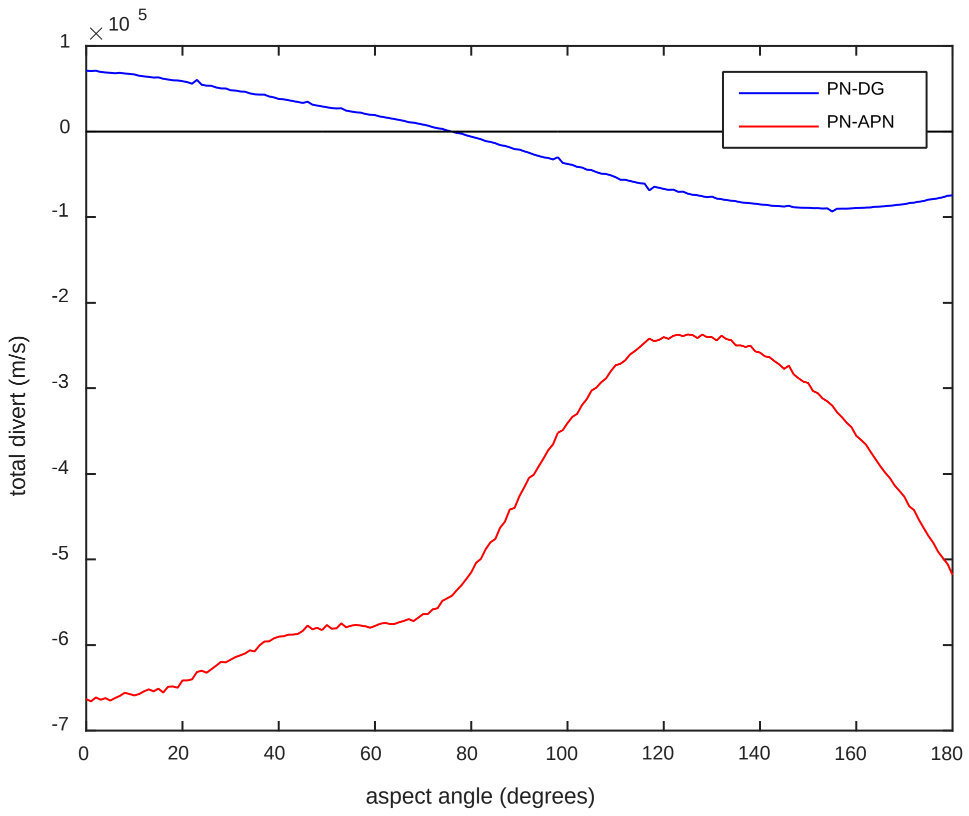


Figure 13. Difference in divert between guidance laws against maneuvering targets at 8000 meters in a noiseless environment.

The difference in the impact time between PN and DG is insignificant. On the other hand, comparing APN to PN and DG shows that APN causes the missile to hit the target 0.35 seconds later, as shown in Figure 14.

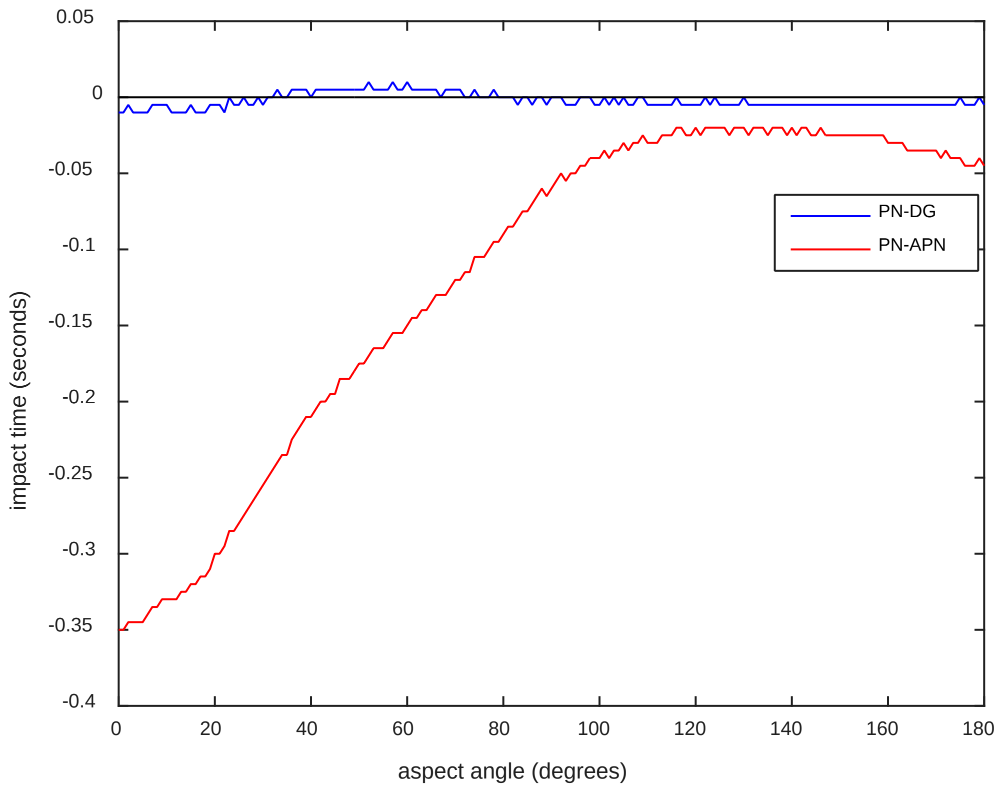


Figure 14. Difference in impact time between guidance laws against maneuvering targets at 8000 meters in a noiseless environment.

As in the divert comparison, DG has better performance than PN and APN as regards impact velocity. As shown in Figure 15, DG provides higher impact velocity to the simulated missile against maneuvering targets with a target heading up to 76 degrees.

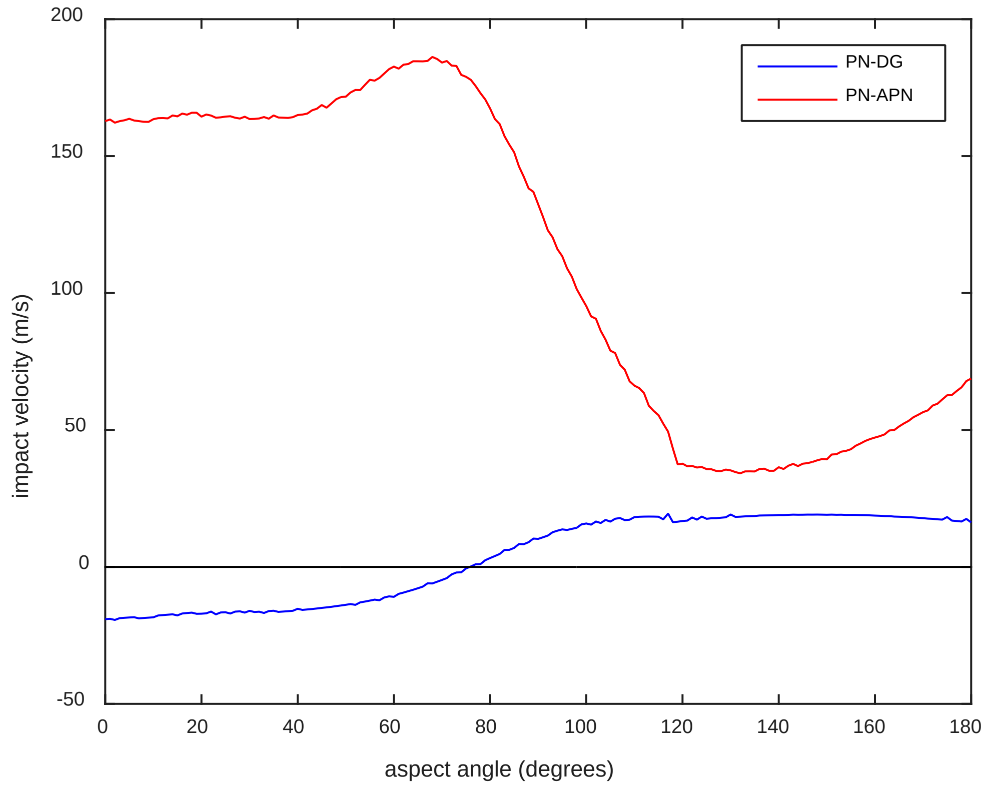


Figure 15. Difference in impact velocity between guidance laws against maneuvering targets at 8000 meters in a noiseless environment.

3. Against Maneuvering Targets at 15000 Meters

In a noiseless environment, using the same parameters for guidance law comparison, it is found that DG performs better against maneuvering targets initially located at 15000 meters and with a target heading up to 71 degrees. In Figure 16, it can be seen that APN is still not competitive in this kind of scenario where a SAM intercepts a target at low altitude.

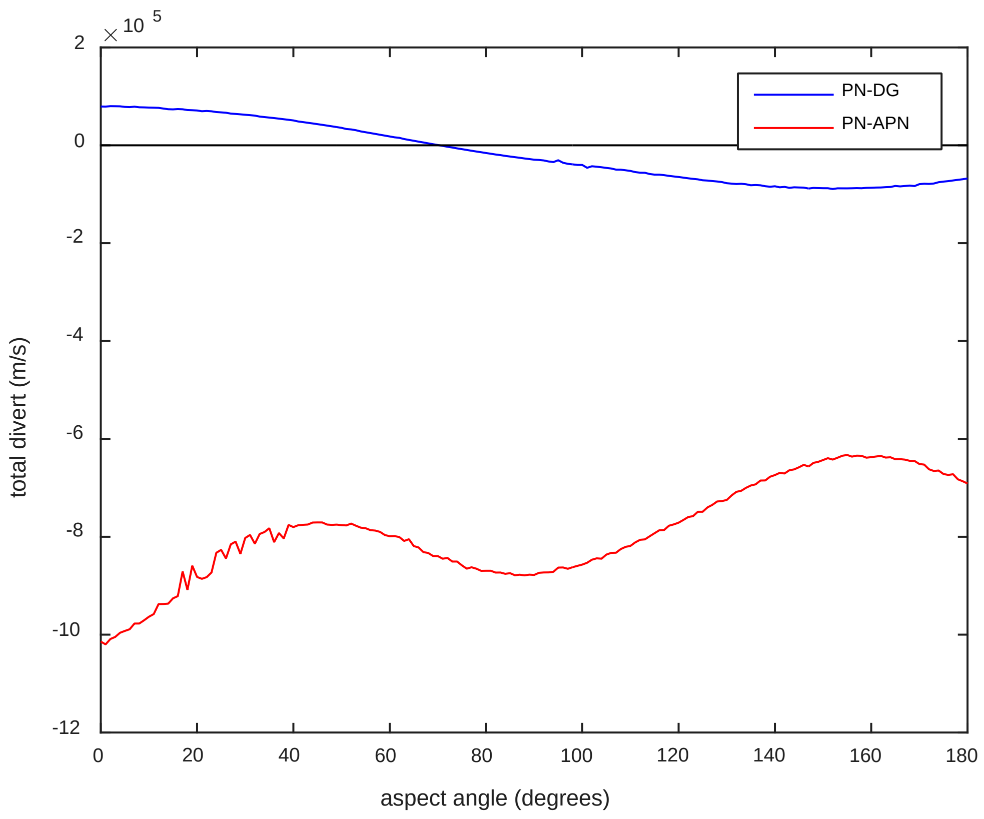


Figure 16. Difference in divert between guidance laws against maneuvering targets at 15000 meters in a noiseless environment.

The impact-time comparison in the 15000 meters scenario gives similar results to the divert comparison results. DG has a lower impact time up to 63 degrees as compared to PN. APN has a higher impact time for all aspects, as shown in Figure 17. A SAM using APN hits the target 0.7 seconds later than a missile using PN guidance. As in the divert comparison, DG has higher impact velocity up to 70 degrees, and APN makes the missile slow too much compared to other guidance laws.

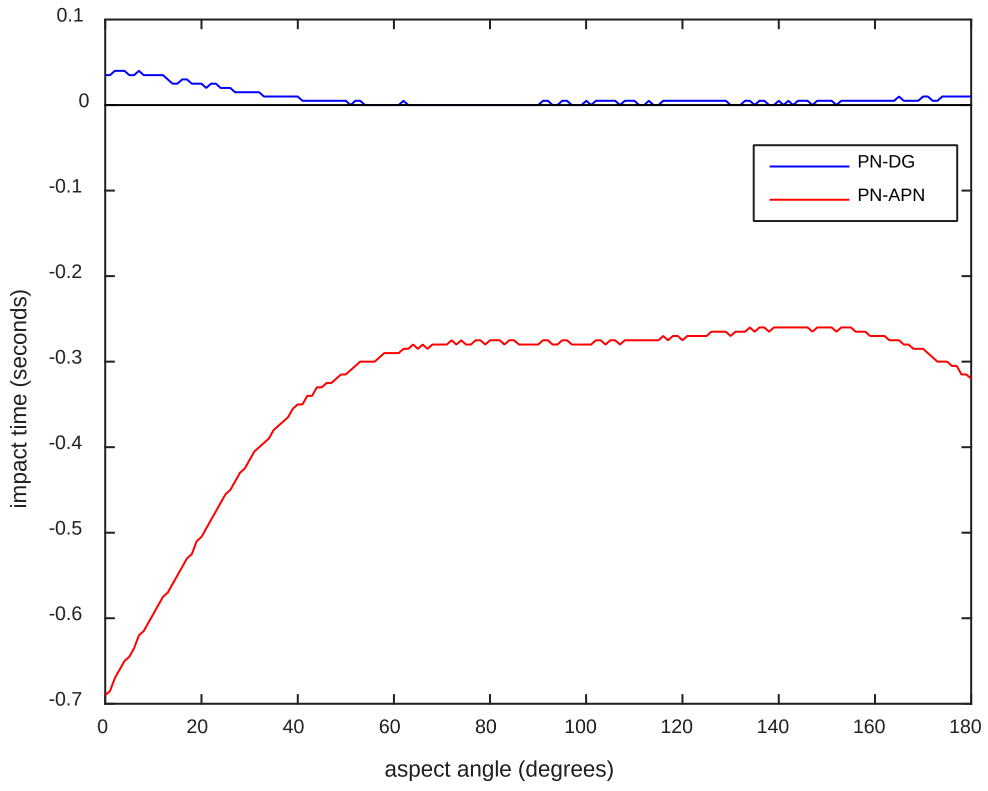


Figure 17. Difference in impact time between guidance laws against maneuvering targets at 15000 meters in a noiseless environment.

B. SIMULATION RESULTS FOR KALMAN FILTERED NOISE STUDY

In this section, the effect of the sensor noise is examined for PN and DG guidance laws. Noisy measurements are filtered using the Kalman filter algorithm described in Chapter II with the modifications presented in Chapter III. In a noiseless environment, all three guidance laws gave similar results against a non-maneuvering target. As seen from the results of previous section, APN performance is unsatisfactory for this type of scenario, where a SAM intercepts a low-level flying CM; therefore, sensor-noise simulations are run against maneuvering targets with different ranges using PN and DG guidance laws. The results are presented in the same manner as in the noiseless environment cases. It is important to note that both PN and DG satisfied a 70% hit probability in all filtered noise simulations with the baseline noise factor equal to one, $f_{noise} = 1$. The vertical axis for the difference curve, PN-DG, appears on the right side of the figures in which the comparison of impact time and impact velocity are shown.

1. Against Maneuvering Targets at 2000 Meters

As in the noiseless simulations, DG tends to have better performance in the aft quadrant. Performance increases for DG starts at 25 degrees and continues up to 165 degrees, as shown in Figure 18. Compared to the noiseless environment results, the performance degradation up to 25 degrees is obvious. Since DG has the estimate of target acceleration magnitude in its formulation, it is affected by noisy estimates. Even when outputs from filters are relatively close to the actual measurements, noisy filter outputs create target acceleration with a magnitude different from zero when the target is not maneuvering. The excessive target acceleration induces extra guidance command and extra drag that slows the missile. The main reason that this excessive guidance command and drag happens in the first 25 degrees is because this tail-chase geometry generates a relatively longer intercept scenario.

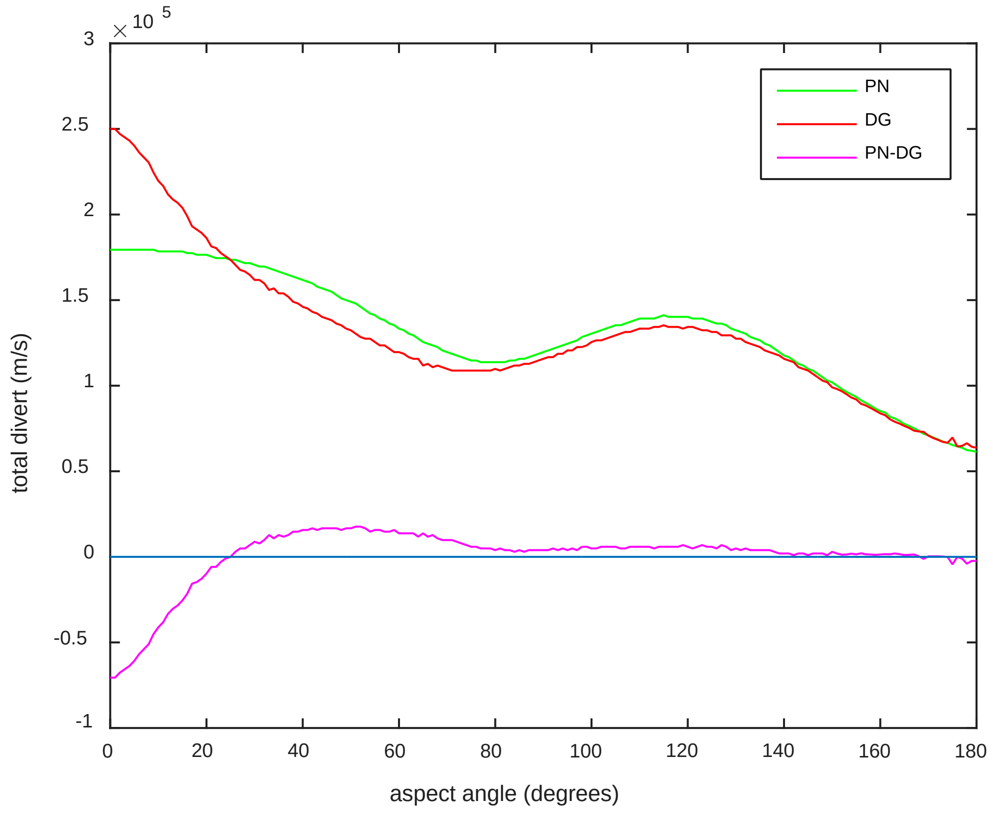


Figure 18. Comparison of divert between PN and DG guidance laws with filtered noise against maneuvering targets at 2000 meters.

It is expected that DG has a shorter impact time between 25–165 degrees. Comparing the difference in impact time between the guidance laws shows that DG has smallest impact time for almost all aspect angles except 160–180 degrees, as shown in Figure 19. The missile is under thrust for all aspect angles in the 2000 meters scenario, and the difference in impact time, 0.11 seconds, is relatively small. It is also expected that PN has a shorter impact time at 0–25 degrees, but since the missile is under thrust, it does not make a significant change in the impact time.

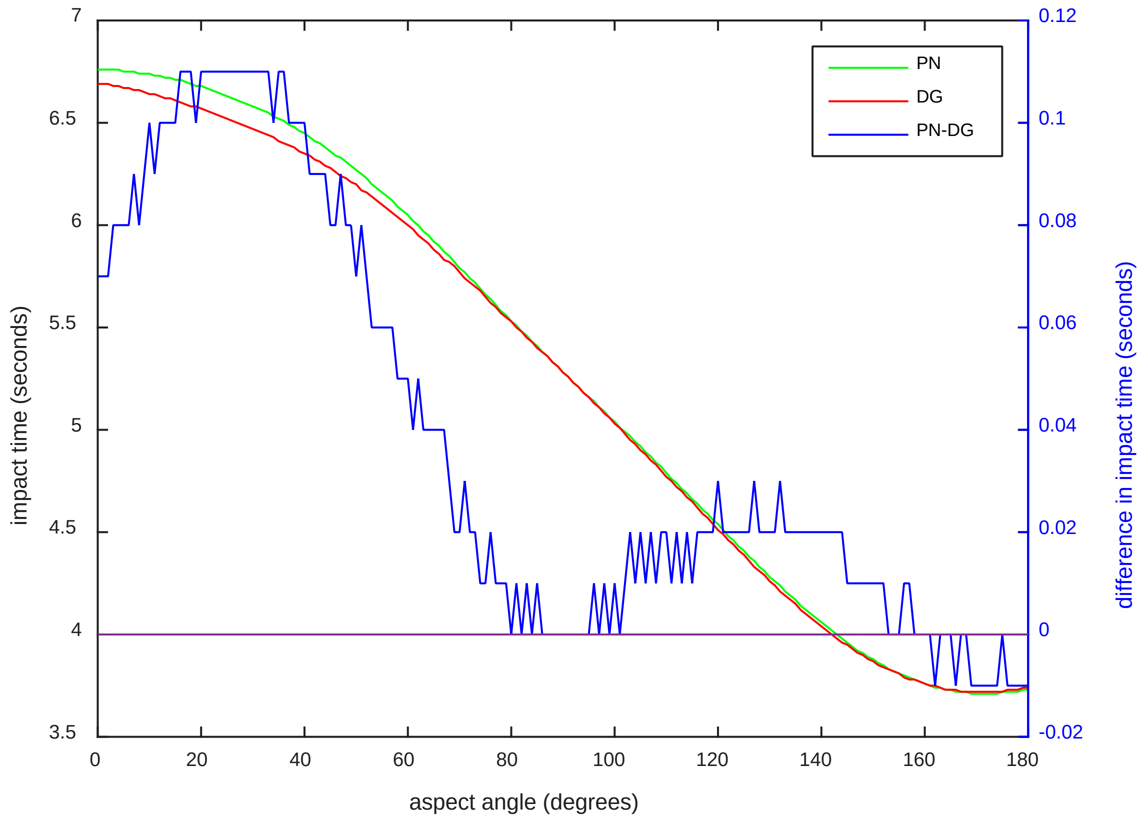


Figure 19. Comparison of impact time between PN and DG guidance laws with filtered noise against maneuvering targets at 2000 meters.

The divert comparison shows that PN has better performance than DG at 0–25 degrees. A larger impact velocity with PN in that region is expected, as shown in Figure 20. The divert difference in other aspect angles is relatively insignificant, and DG has no advantage in providing a larger terminal missile speed.

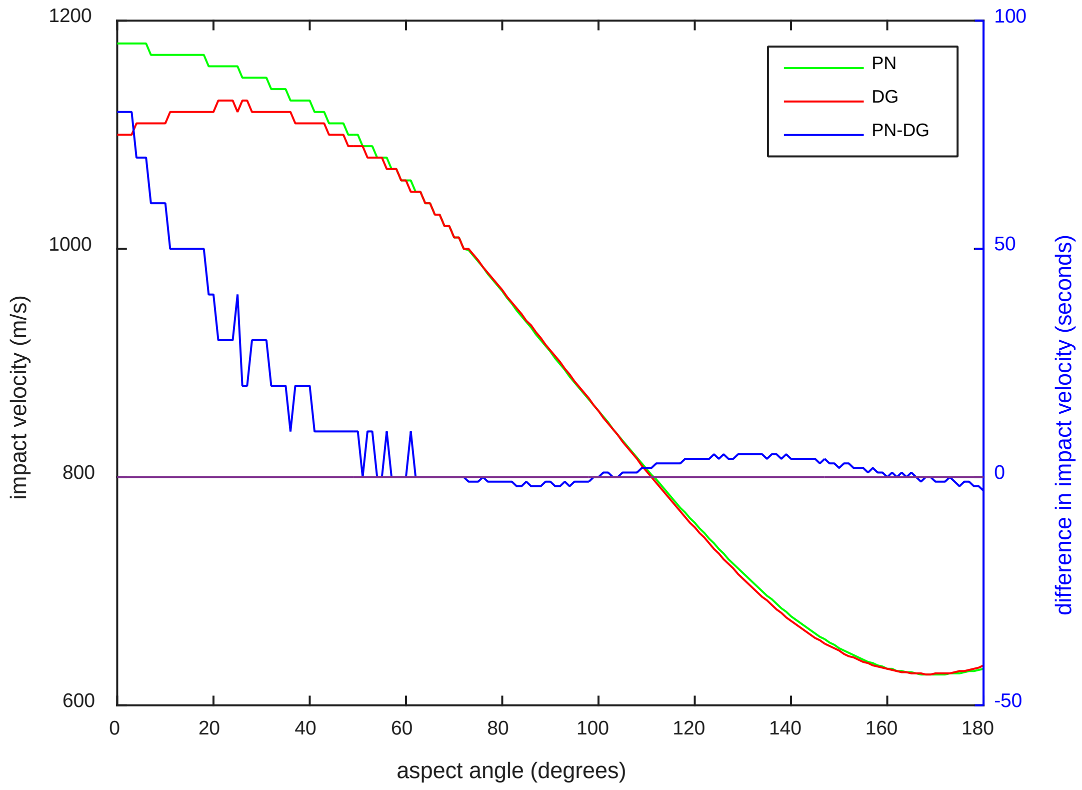


Figure 20. Comparison of impact velocity between PN and DG guidance laws with filtered noise against maneuvering targets at 2000 meters.

2. Against Maneuvering Targets at 8000 Meters

Even though there is performance degradation up to 9 degrees, DG still performs better in the aft and broadside quadrants between 10–109 degrees, as shown in Figure 21. Excessive target acceleration estimates produce unnecessary guidance command and drag. Performance degradation caused by extra drag can be seen at 0–9 degrees.

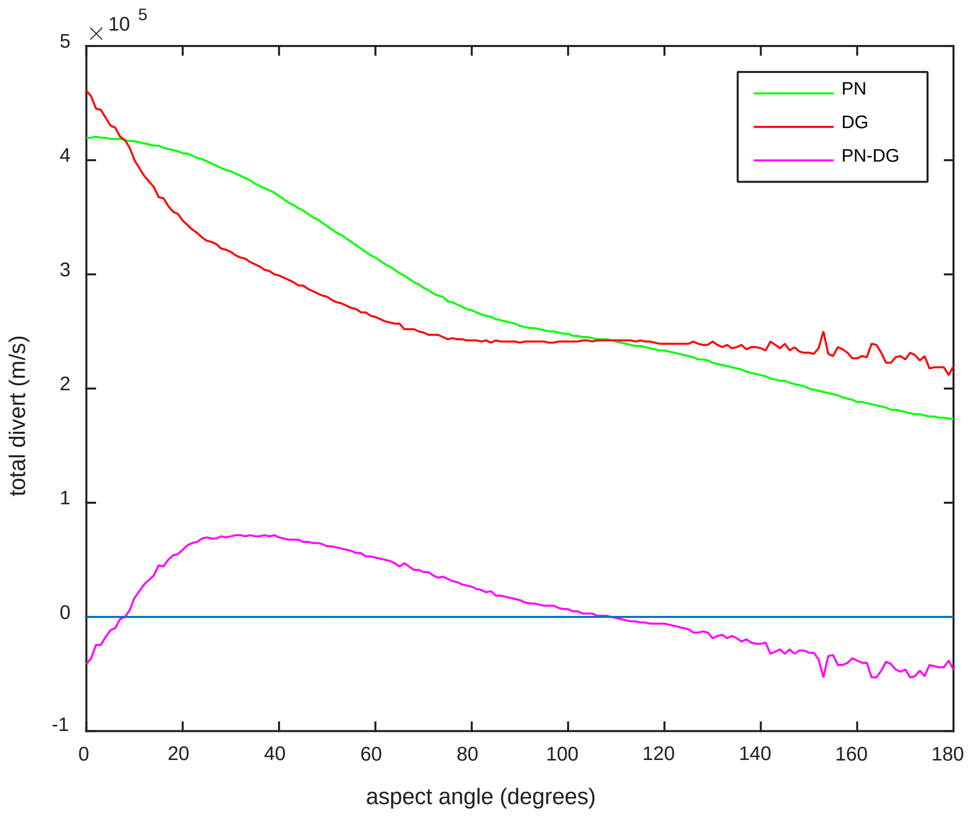


Figure 21. Comparison of divert between PN and DG guidance laws with filtered noise against maneuvering targets at 8000 meters.

As seen in the results of simulations with noiseless measurements, DG performs better in tail-chase scenarios against maneuvering targets at 8000 meters by intercepting in a shorter time. It can be seen in Figure 22 that DG has a shorter impact time between 0–97 degrees.

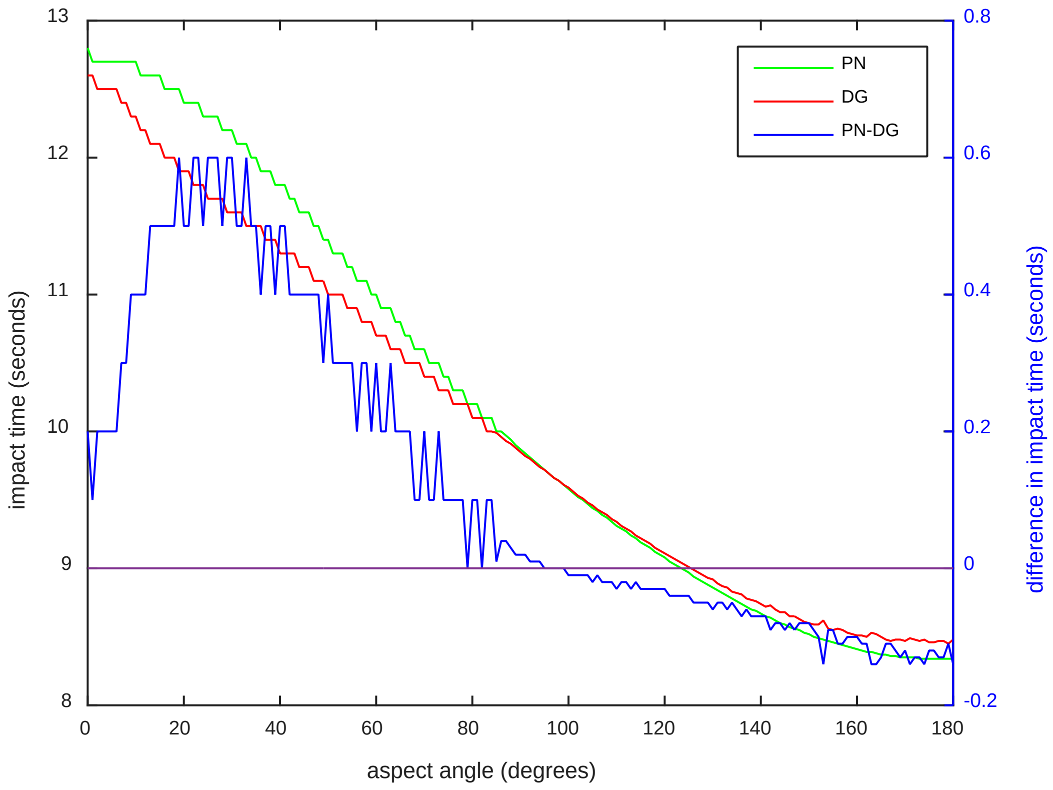


Figure 22. Comparison of impact time between PN and DG guidance laws with filtered noise against maneuvering targets at 8000 meters.

In the divert comparison for 8000 meters, DG has a performance loss between 0–9 degrees. Compatible with the divert comparison results, the impact velocity difference between the guidance laws shows that DG provides higher impact velocity for aspect angles greater than 14 degrees, as shown in Figure 23. It should be noted that in stern chase and broadside aspects (angles<115 degrees), the missile transitions from thrust to coast while under active guidance. As a result, greater divert implies greater drag and absence of thrust directly degrades impact velocity in stern chase scenarios, while forward aspect scenarios, always under thrust, are not significantly affected.

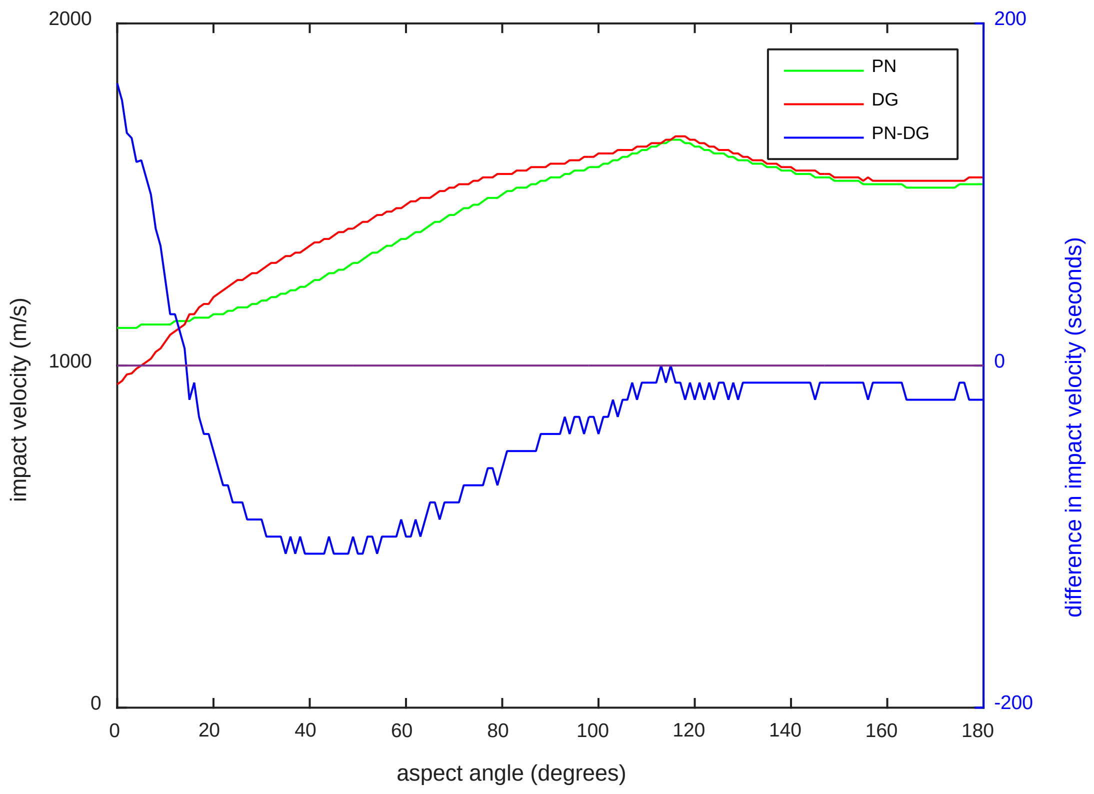


Figure 23. Comparison of impact velocity between PN and DG guidance laws with filtered noise against maneuvering targets at 8000 meters.

3. Against Maneuvering Targets at 15000 Meters

Performance degradation for DG is evident up to 40 degrees against maneuvering targets at 15000 meters. DG shows better performance only between 40–76 degrees. At aspect angles greater than 76 degrees, PN requires less divert.

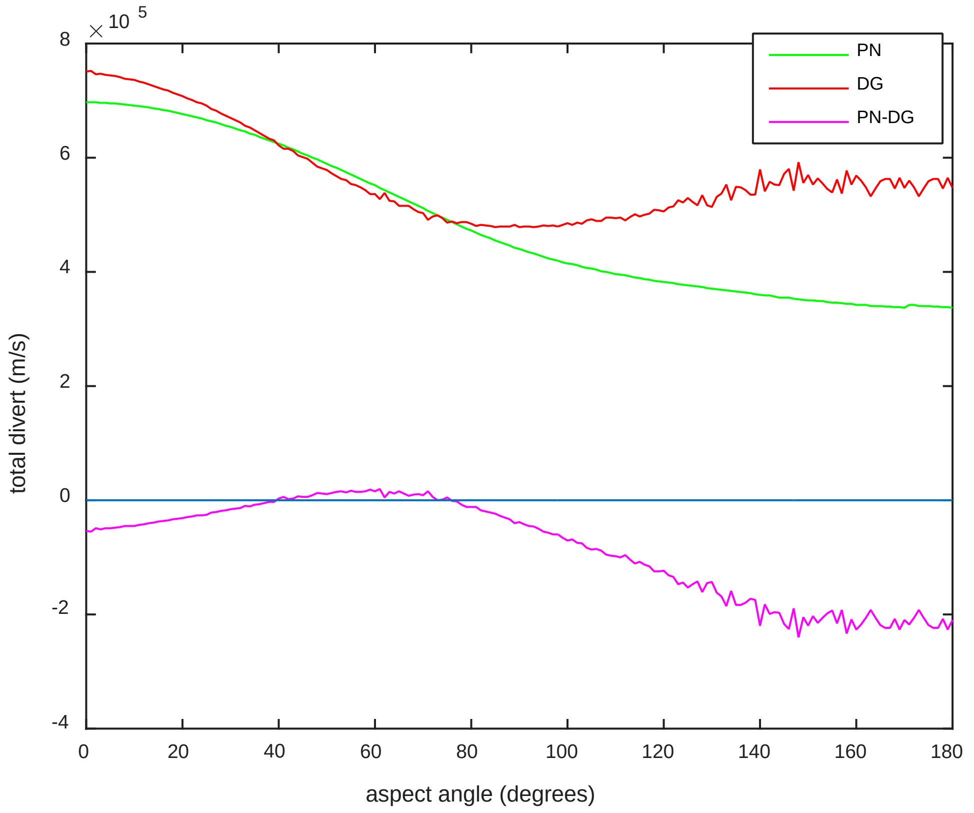


Figure 24. Comparison of divert between PN and DG guidance laws with filtered noise against maneuvering targets at 15000 meters.

Compatible with the results from divert comparison, the impact time of DG shows better results between 33–64 degrees. Except in this region, PN has shorter impact time compared to DG, as shown in Figure 25. The observed value of 1.5 seconds of impact time difference is a significant advantage of PN in this scenario.

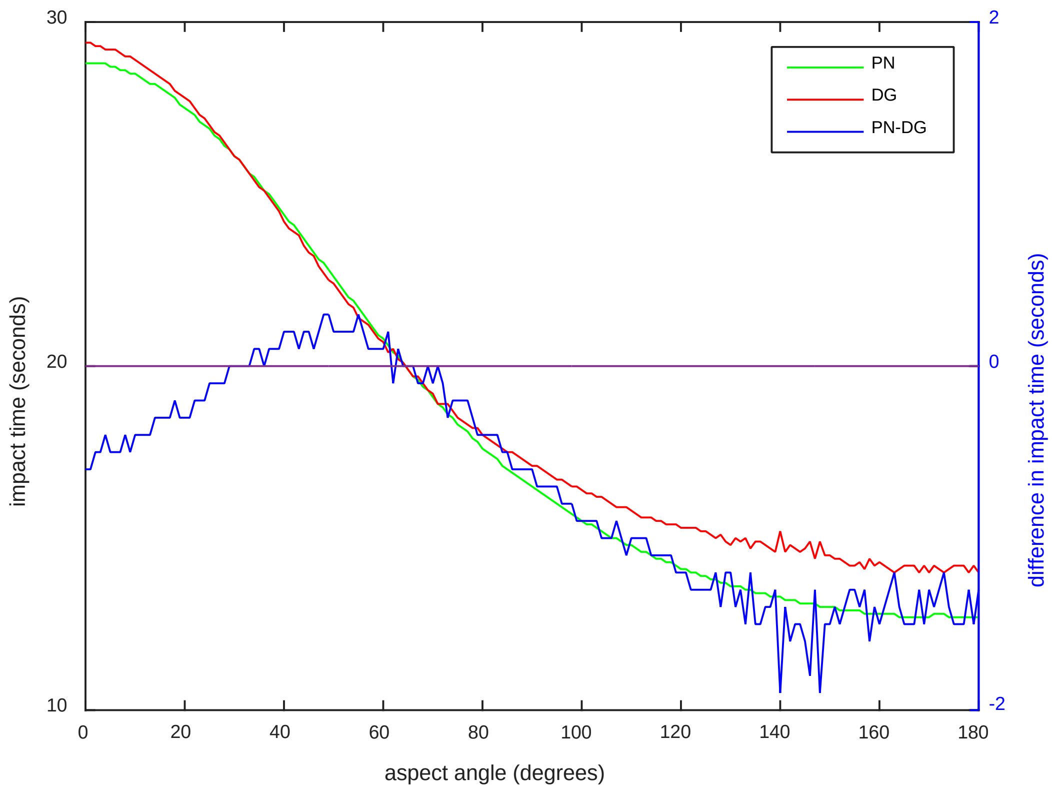


Figure 25. Comparison of impact time between PN and DG guidance laws with filtered noise against maneuvering targets at 15000 meters.

Although DG can provide slightly higher impact velocity between 46–72 degrees, PN performs better at most of the aspect angles, as shown in Figure 26. The difference in impact velocity can be up to 246 m/s, and this is a significant advantage of PN in this scenario. It should be noted that in all of these long-range scenarios, the missile transitions from thrust to coast for all aspect angles. Absence of thrust degrades the impact velocity for all aspect angles.

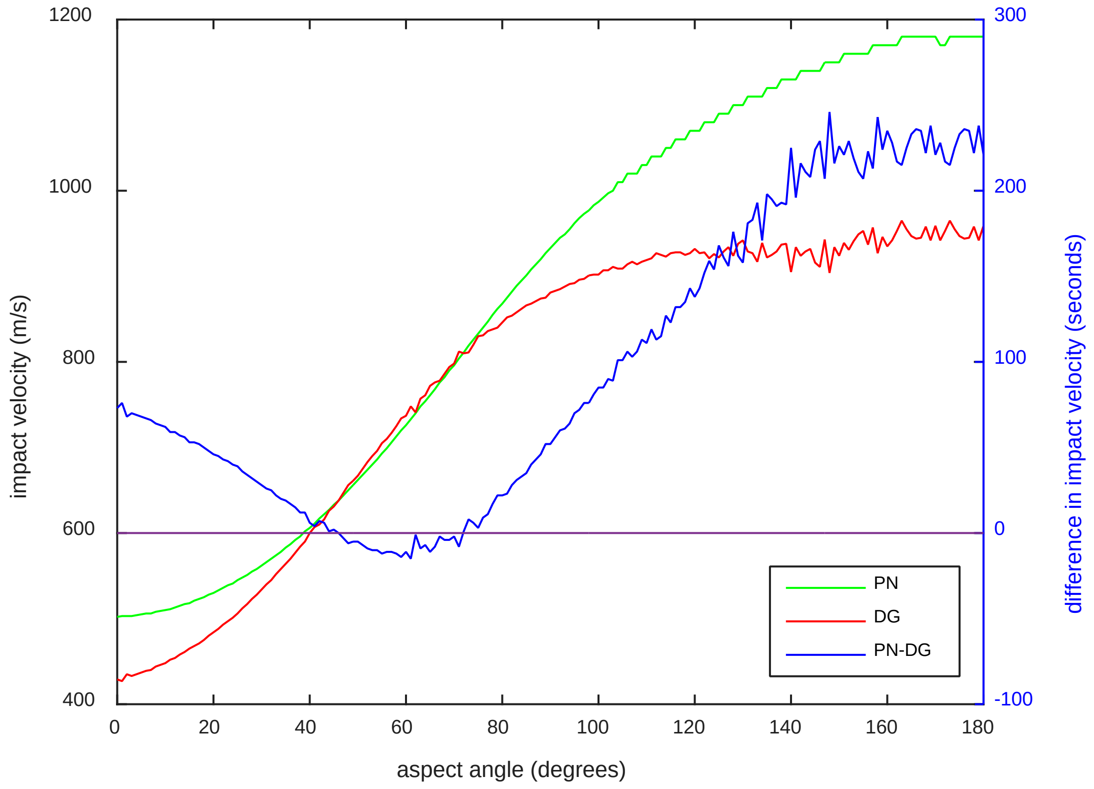


Figure 26. Comparison of impact velocity between PN and DG guidance laws with filtered noise against maneuvering targets at 15000 meters.

C. MAXIMUM NOISE TOLERANCE

Noise tolerance is measured by determining the maximum noise factor that each guidance law can withstand while still maintaining 70% hit probability. The maximum value of the noise factor is determined using iterative increases of the noise factor multiplier in equations (4.1), (4.5), and (4.6) until the probability of a hit falls below 70%, based on 50 scenario iterations. PN and DG guidance laws are tested from 0 to 180 degrees of aspect angle, with a one degree interval, against maneuvering targets at 2000 meters and 8000 meters.

As seen from the maximum noise factor result for 2000 meters, DG can tolerate more noise up to 100 degrees. DG is still better in the aft quadrant except 0–20 degrees. Also, DG has a higher noise tolerance starting from 135 degrees up to 180 degrees, as shown in Figure 27. We can say that DG can withstand more noise than PN for short range, 2000 meters.

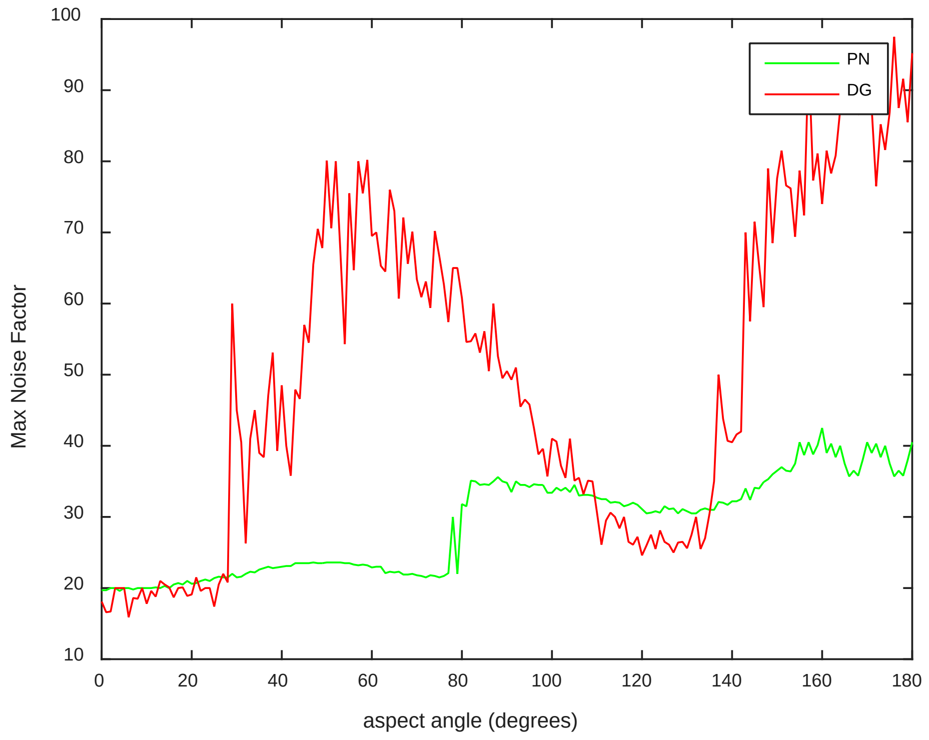


Figure 27. Maximum noise factor test for PN and DG against maneuvering targets at 2000 meters with 50 runs.

Earlier results of guidance law comparisons with Kalman filtered noise at 8000 meters indicate that DG has better performance in the region of 10–109 degrees of aspect angle. Except for this region, PN provides better results with less divert, shorter impact time, and higher impact velocity. Compatible with these earlier results, DG has higher noise tolerance up to 85 degrees of aspect angle, and PN has the noise factor advantage at aspect angles greater than 145 degrees. The peak region between 105 and 145 degrees, where the maximum noise factor of DG increases up to 70, is shown in Figure 28. It is important to note that a missile is under continuous thrust in this region of aspect angles but not at lower angles. It is unclear why there is an abrupt noise tolerance change for DG at forward aspect angles greater than 145 degrees.

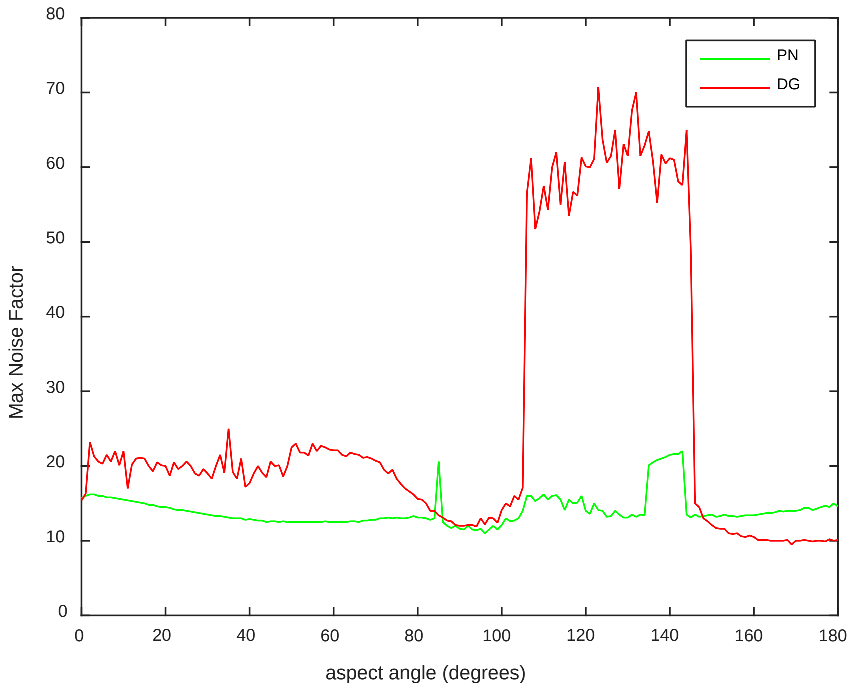


Figure 28. Maximum noise factor test for PN and DG against maneuvering targets at 8000 meters with 50 runs.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

The objectives of this research were achieved. A three-dimensional simulation model was designed to intercept a CM flying at low altitude at constant speed. The missile and target were simulated using real PAC-3 and Tomahawk cruise missile characteristics obtained from open-source literature.

For the performance analysis of guidance laws, divert, impact time, and impact velocity were used as measures of effectiveness. The analysis showed that these parameters are related to each other and results were consistent. When divert was smaller, impact time was shorter and impact velocity higher, as expected.

At first the guidance laws were examined with ideal measurements to see the behavior in three-dimensional intercept scenarios. It is important to note that at the simulated ranges, all three guidance laws achieved 100% hit probability against maneuvering and non-maneuvering targets using noiseless measurements. Against non-maneuvering targets at all three ranges, all guidance laws performed the same. The additional target related terms in APN and DG did not take effect, since for a non-maneuvering target the magnitude of target acceleration was zero if actual measurements were used.

Against maneuvering targets, the DG guidance law performed better in tail-chase scenarios at three different ranges under noiseless conditions. In the aft quadrant, DG required smaller divert and provided a target hit with shorter impact time and higher impact velocity compared to PN and APN. Although APN succeeded in hitting the target in all cases, it performed poorly compared to the other guidance laws in these scenarios; therefore, APN was taken out of the comparison in the Kalman filtered noise study.

Under noisy conditions, PN and DG were examined using the performance parameters. Guidance laws were tested against maneuvering targets from all aspect angles. Using the filtered measurements, we saw that DG showed similar behavior in the aft quadrant, requiring smaller divert, providing a shorter impact time, and providing higher impact velocity. For tail-chase scenarios, DG gave better results except for the first ten to 25 degrees of aspect angle of outgoing targets. The performance degradation happened because of the estimate of the target acceleration magnitude term in its formulation. This term is computed from the filter outputs. The filters generated the parameter estimates close to the actual ones, but small differences between the filter outputs and actual parameters caused unnecessary target acceleration. The generated target acceleration while the target was not maneuvering induced unnecessary guidance commands and drag. These factors made the missile slow down in the first ten to 25 degrees of aspect angles. The degradation caused by producing extra guidance and drag due to de facto target acceleration occurred in all ranges. The performance degradation expanded up to 40 degrees of aspect angle as the range increased.

Maximum noise factor tests for 2000 meters and 8000 meters showed that DG has higher noise tolerance. DG showed better performance using the analysis parameters in the filtered study part. The higher noise factor values in the aft quadrant for DG match this result. DG and PN had no significant difference in the forward quadrant against targets at 2000 meters, but DG can tolerate more noise than PN in the forward quadrant at 2000 meters. Compatible with the results for 8000 meters, the maximum noise factor for PN was greater after 145 degrees of aspect angle. The explanation for the peak region of the maximum noise factor for DG is unclear and needs further investigation.

B. RECOMMENDATIONS FOR FURTHER WORK

1. Enhanced Filtering Techniques

The main instability in the results of this research was caused by the parameters generated from Kalman filter outputs; therefore, more advanced filters may mitigate the noise effects and generate better estimates. Enhanced filtering techniques, such as an interacting multiple model (IMM), may provide superior results.

2. Studying Different Guidance Laws

Guidance for the tactical missiles is increasingly complex in order to improve accuracy and hit probability. Tactical missiles, which are designed to intercept CMs, are usually directed to the predicted intercept point calculated before the launch. This point is always updated either by the ground radar or the missile's onboard guidance computer. A guidance law based on the predicted intercept point can be more complicated but might improve the tactical performance of a SAM against a cruise missile. Also, a guidance law based on the angle of impact can be examined in the terminal homing phase. Determination of the angle of impact, which depends on closing speed, altitude, and target acceleration, can increase the lethality of the missile.

3. Random Target Maneuvers

In the simulations, the target started a constant 6.89 g turn three seconds prior to the impact. In reality, every CM has its own characteristics for evasive maneuver. The target turn initiation time can be implemented between one and four seconds using a random number generator. Also, the magnitude of the target turn can be generated randomly to examine the hit probability of the guidance laws.

4. Drag Model and Acceleration Limiter Improvement

In the simulations, parasitic drag and induced drag were implemented in a simple way using a similar model close to the empirical drag model for missiles. More realistic aerodynamics analysis of a missile can be done using software, such as Missile DATCOM supplied by the U.S. Air Force. This type of software calculates the center of pressure, center of gravity, and the drag coefficients of a missile for all AOAs. A 6DOF model uses roll, pitch, and yaw angles. Induced drag is a function of AOA. Drag can be calculated with these angles and the parameters obtained from the software.

Additionally, roll, pitch, and yaw angles should be limited in a real scenario. In the simulation, guidance acceleration is limited using a constant value not to exceed the missile's lateral acceleration capability. Instead, the angles used in the 6DOF models can be limited to accurately represent the missile's maneuverability.

5. Parallel Computing

The computation time to examine the difference between the guidance laws took one to five hours, depending on the range. Finding the maximum noise factor with 50 iterations for all aspect angles took greater than 100 hours. To decrease the computation time, code generation for the simulation can be designed to be compatible with parallel computing. A super computer, such as Hamming at the Naval Postgraduate School, can use multiple cores at the same time. Faster simulation results can be obtained using parallel computing. Also, running 100 iterations for the maximum noise factor comparison can give better statistical results.

APPENDIX. MATLAB CODE

Some portions of Osborn's [14] MATLAB codes were used and modified. All the MATLAB script files used in this thesis are presented in Table 3.

Table 3. List of the MATLAB script files used in this study.

FILENAME	PURPOSE
A. Simulation run script files	
init.m	Initializes the global variables.
MAIN.m	Runs the desired simulation.
SIM.m	Runs the simulation script.
NF_max.m	Tests the guidance laws for maximum noise factor.
B. Simulation guidance law files	
PN_GL.m	Implements the proportional navigation guidance law.
DG_GL.m	Implements the differential geometry guidance law.
APN_GL.m	Implements the augmented proportional navigation guidance law.
C. Simulation function files	
time_to_impact.m	Calculates time remaining before impact with the target.
mach_speed.m	Calculates the Mach number of missile speed.
mass.m	Calculates the missile's mass depending on the simulation time.
rho_value.m	Calculates the density of air at a given altitude.
cdp_value.m	Calculates the parasitic drag coefficient from Figure 5.
fdp_value.m	Calculates the force on the missile due to parasitic drag.
noisy_data.m	Adds noise to LOS angles and range measurements.
missile_motion.m	Updates the missile state vector.
target_motion.m	Updates the target state vector.
divert.m	Calculates divert after guidance flag turns to 1.
D. Simulation filter files	
kalman_pn_range.m	Generates corrected estimates of the range state for the proportional navigation guidance law.
kalman_pn_theta.m	Generates corrected estimates of the LOS angle state for the proportional navigation guidance law.
kalman_dg_range.m	Generates corrected estimates of the range state for the differential geometry guidance law.

A. SIMULATION RUN SCRIPT FILES

```
% INIT
%-----------------------------------------------------------------------
%   File:               init.m
%   Name:               1st LT Murat DOGEN
%   Component Runtime:  8.5.0.197613 (R2015a)
%   Compiler:           6 (R2015a)
%                       64-bit (Windows 8.1)
%   Date:               06 June 2015
%   Description:        Generates global variables in the workspace.
%                       Allows user to modify initialization parameters
%-----------------------------------------------------------------------
global TargetTurnG  tspeed delta t_impact Nprime stoptime maxg time...
GRAV  DIAM SREF thrust burntime mass_total mass_impact alt_T...
SIG_RNG SIG_THETA

%SIMULATION
stoptime = 50;  % Maximum scenario run time (sec).
delta = 0.005;  % Discrete step size (sec).
t_impact=15;    % initial impact time value
GRAV=9.8045;    % Gravitational constant (m/sec^2).
time = 0;       % Initial simulation time (sec).
%FILTER
SIG_RNG      = NZ_FACTOR*10;    % Range sensor uncertainty (m).
SIG_THETA    = NZ_FACTOR*0.001; % LOS angle sensor uncertainty (rad).
Nprime=5;   % Effective navigation ratio.
maxg = 50;  % Maximum Guidance Acceleration (g).
%MISSILE
DIAM=0.255; % Diameter of the simulated missile
SREF=pi*DIAM^2/4;   % Missile cross sectional area (m^2)
mass_total=315;     % Mass of the simulated missile before launch (kg).
mass_impact=142;    % Mass of the simulated missile at impact time
(kg).
thrust_to_weight=15.57; % Thrust-to-weight ratio of the simulated
missile.
thrust=mass_total*thrust_to_weight*GRAV; % Thrust (Newton).
Isp=260;            % Specific impulse time of the simulated missile
(sec).
burntime=Isp*(mass_total-mass_impact)*GRAV/thrust; % Missile burn time
(sec).
%TARGET
TargetTurnG=6.89;   % Acceleration of target turn (g).
tspeed = 249.6312 ; % Target acceleration (m/s).
alt_T=276.7584; % Target altitude-constant (m).
```

```
% MAIN
%----------------------------------------------------------------------
%----------------------------------------------------------------------
%   File:               MAIN.m
%   Name:               1st LT Murat DOGEN
%   Component Runtime:  8.5.0.197613 (R2015a)
%   Compiler:           6 (R2015a)
%                       64-bit (Windows 8.1)
%   Date:               06 June 2015
%   Description:        Runs the desired simulation
%----------------------------------------------------------------------
clear all;close all;format shorte
format longe
tic,
iter = 20;  % number of iterations/runs
            % for noise study:20
            % noiseless :1
%select target distance (km)
D_target = 2;
% D_target = 8;
% D_target = 15;
global  ANG
ANG = 0:1:180; % aspect angles
TURN = 1; % Set 1 for maneuvering target
NZ = 0;NZ_FACTOR = 0;FILT = 0; % Noiseless study
% NZ = 1;NZ_FACTOR = 1;FILT = 1; % Noise Study
init ; % installing global parameters

% initial matrices
HIT=zeros(length(ANG),1);
HIT2=zeros(length(ANG),1);
HIT3=zeros(length(ANG),1);
RESULT1=zeros(5,length(ANG));
RESULT2=zeros(5,length(ANG));
RESULT3=zeros(5,length(ANG));

for aa=1:max(size(ANG))
    for ii=1:iter
        %initiate target state vector
        Ti = [  D_target*1000 tspeed*cosd(ANG(aa))...
                0 tspeed*sind(ANG(aa)) ...
                -alt_T 0]';

        guidance_law = 1 ; SIM;
        if RES(4)>= 5
            disp( ['ITER ',num2str(ii),...
                ' angle ' num2str(ANG(aa)),...
                '  divert: ',num2str(RES(1)),...
                ' !!!! MISS !!!! ', num2str(RES(4)),...
                ' MissileSpeed=  ',num2str(vm)])
        else
            disp( ['ITER ',num2str(ii),...
                ' angle ' num2str(ANG(aa)),...
                '  divert: ',num2str(RES(1)),...
                ' hit  ', num2str(RES(4)),...
                ' MissileSpeed=  ',num2str(vm)])
            HIT(aa)=HIT(aa)+1;
            RESULT1(:,aa)=RESULT1(:,aa)+RES;
        end

        guidance_law = 2 ;SIM;
        if RES(4)>= 5
            disp( ['ITER ',num2str(ii),...
                ' angle ' num2str(ANG(aa)),...
                '  divert: ',num2str(RES(1)),...
                ' !!!! MISS !!!! ', num2str(RES(4)),...
                ' MissileSpeed=  ',num2str(vm)])
        else
            disp( ['ITER ',num2str(ii),...
                ' angle ' num2str(ANG(aa)),...
                '  divert: ',num2str(RES(1)),...
                ' hit  ', num2str(RES(4)),...
                ' MissileSpeed=  ',num2str(vm)])
            HIT2(aa)=HIT2(aa)+1;
            RESULT2(:,aa)=RESULT2(:,aa)+RES;
        end

        guidance_law = 3 ;SIM;
        if RES(4)>= 5
            disp( ['ITER ',num2str(ii),...
                ' angle ' num2str(ANG(aa)),...
                '  divert: ',num2str(RES(1)),...
                ' !!!! MISS !!!! ', num2str(RES(4)),...
                ' MissileSpeed=  ',num2str(vm)])
        else
            disp( ['ITER ',num2str(ii),...
                ' angle ' num2str(ANG(aa)),...
                '  divert: ',num2str(RES(1)),...
                ' hit  ', num2str(RES(4)),...
                ' MissileSpeed=  ',num2str(vm)])
            HIT3(aa)=HIT3(aa)+1;
            RESULT3(:,aa)=RESULT3(:,aa)+RES;
        end
    end
    %averaging the results for each guidance law
    R1(:,aa)= ( RESULT1(:,aa) /HIT(aa));
    R2(:,aa)= ( RESULT2(:,aa) /HIT2(aa));
    R3(:,aa)= ( RESULT3(:,aa) /HIT3(aa));
end
toc;
```

```
% SIM
%-----------------------------------------------------------------------
%-----------------------------------------------------------------------
%   File:               SIM.m
%   Name:               1st LT Murat DOGEN
%   Component Runtime:  8.5.0.197613 (R2015a)
%   Compiler:           6 (R2015a)
%                       64-bit (Windows 8.1)
%   Date:               06 June 2015
%   Description:        Runs the simulation script.
%-----------------------------------------------------------------------
%% Initial Parameters
vla=15+(D_target-2)*2;  %VERTICAL launch angle
guidance_flag=0;        % initial set for guidance flag
Divert=[0;0];           % initial set for divert matrix
time = 0;               % initial set of simulation time
%%
T=Ti;   % Initialize Missile State Vector:  [x vx y vy z vz]';
Vi=eps; % initial launch speed
laz=atan2(Ti(3),Ti(1));%launch angle towards the target(rad)
%%
Vi=[Vi*cos(laz)*cosd(vla) Vi*sin(laz)*cosd(vla) -Vi*sind(vla)];
Mi= [0 Vi(1)  0 Vi(2)  0 Vi(3)]'; % initial missile state vector
M=Mi;
%Matrixes to extract position and velocity components
Hp = [1 0 0 0 0 0;0 0 1 0 0 0;0 0 0 0 1 0];  %position
Hv = [0 1 0 0 0 0;0 0 0 1 0 0;0 0 0 0 0 1];  %velocity
% Initial LOS range and angles
Vt = Hv*T;       % target Velocity Vector
Vm = Hv*M;       % Missile Velocity Vector
vm = norm(Vm);   % Missile Speed
Vm_u=Vm/vm;      % unit vector of Vm
Range = Hp*(T-M);% LOS Vector pt-pm
range=norm(Range); % range to target
Range_u=Range/range;% unit r vector
theta_az=( atan2( Range(2),Range(1)));
theta_el= atan2( Range(3),norm(Range(1:2))) ;
theta_az_dot=0;
theta_el_dot=0;
%% Initial thrust, drag, guidance acc values
mab=thrust/mass_total/GRAV;%in g
Mburn=mab*GRAV*Vm/vm;burn=1; %in m/s^2mag=0;
Mag=zeros(3,1);
Fdp=0; % parasitic drag
Fdi=0;
mad=(Fdp+Fdi)/mass(time);% ==0 in m/s^2
%% Initial filter input outputs
mac_paraL_k1=-dot(Mburn,Range_u);%in m/s^2
mac_perpL_k1=mab+mac_paraL_k1;
mac_perpL1_k1 = 0;
mac_perpL2_k1 = 0;

Ma=Mburn;% total missile acceleration (m/s^2)
thetaLM_k0 = 0; % missile lead angle previous time estimate
```

```
thetaLM_k1 = 0; % missile lead angle current time estimate
VM_old = Hv*M;
%% initial variables for guidance implementation
Vr=Vt-Vm;  %range rate vector
rangedot=dot(Vr,Range_u);%range rate
Vc=-dot(Vr,Range_u)*Range_u; % closing velocity
vc=-dot(Vr,Range_u); % closing speed (m/s)
Vp=Vr+Vc;
thetadot=norm(Vp)/range; % LOS rate
thetadot_old=norm(Vp)/range;
rangedot_old = 0;
%% INITIAL OUTPUT MATRICES
Range_out = [];     % Relative Position Output Matrix
PoutM = []; % Missile position output matrix
PoutT = []; % Target position output matrix
Vm_out = [];   % Missile Velocity Output Matrix
vm_out = [];   % Missile Speed
Vtout=[];
range_out = [];  % Range
theta_az_out = []; % LOS azimuth
theta_el_out = []; % LOS elevation
theta_az_dot_out = [];% LOS azimuth rate
theta_el_dot_out = [];% LOS elevation rate
acc_out = [];      % total acc
mag_out=[];  % guidance acc
mad_out=[];  % drag acc
thetadot_out=[]; % LOS rate
v_c_out=[]; % closing velocity
tout = [];  % simulation time
t_impact_out = []; % time to go
mburn_out=[]; % thrust acc
map_out=[]; % parasitic drag
mai_out=[]; % induced drag
mach_out=[]; % Mach number
Cdp_out=[]; % drag coeff.
Acc_t_k1_out=[]; % estimate of target acc
 %% MAIN LOOP
 for kk = 1:(stoptime/delta)
%%
malt = M(5);
Vt = Hv*T;  % target Velocity Vector
Vm = Hv*M;  % Missile Velocity Vector
vm = norm(Vm);% Missile Speed
Vm_u = Vm/vm; % unit vector of Vm
vt = norm(Vt); % target speed
Range = Hp*(T-M); % LOS Vector pt-pm
range = norm(Range); % range(m)
Range_u = Range/range;% unit r vector
Vr = Vt-Vm;
rangedot_old = rangedot;
rangedot = dot(Vr,Range_u);
Vp=Vr-rangedot*Range_u;
Vp_u=Vp/norm(Vp);
```

```
% ACTUAL PARAMETERS
theta_az_old=theta_az;
theta_az_dot_old=theta_az_dot;
theta_az=( atan2( Range(2),Range(1)));
theta_az_dot=-(theta_az_old-theta_az)/delta;
theta_az_dotdot=-(theta_az_dot_old-theta_az_dot)/delta;

theta_el_old=theta_el;
theta_el= atan2( Range(3),norm(Range(1:2))) ;
theta_el_dot=(theta_el-theta_el_old)/delta;
theta_el_dot_old=theta_el_dot;
theta_el_dotdot=(theta_el_dot-theta_el_dot_old)/delta;

thetaLM = dot(Range_u,Vm_u);
thetaLT=dot(Range_u,Vt/norm(Vt));

thetadot_old = thetadot;
thetadot=norm(Vp)/range% Actual LOS angle rate
thetadotdot=(thetadot-thetadot_old)/delta;%theta doubledot

rangedotdot=(rangedot-rangedot_old)/delta;%range acc

%% NOISE AND FILTERING SECTION
% Generate noisy measurements
if NZ == 0
    nz_range = range;
    nz_theta_az=theta_az;
    nz_theta_el=theta_el;
else % Noise "ON"

[nz_range,nz_theta_az,nz_theta_el]=noisy_data(range,theta_az,theta_el);
end

% PRE-FILTERING
if FILT==1
    if kk == 4
        range_k0=range_out(1,3);
        rangedot_k0 = (range_out(1,3) - range_out(1,2))/delta;
        rangedotdot_k0=(((range_out(1,3) - range_out(1,2))/delta)-...
                        ((range_out(1,2) -
range_out(1,1))/delta))/delta;

        theta_az_k0=theta_az_out(1,3);
        theta_az_dot_k0 = theta_az_dot_out(1,3);
        theta_az_dotdot_k0 = (theta_az_dot_out(1,3)-...
                                theta_az_dot_out(1,3))/delta;

        theta_el_k0=theta_el_out(1,3);
        theta_el_dot_k0 = theta_el_dot_out(1,3);
        theta_el_dotdot_k0 = (theta_el_dot_out(1,3)-...
                              theta_el_dot_out(1,3))/delta;
        if guidance_law == 1 % PN
            Prng_k0 = SIG_RNG^2*diag([1;(2/(delta^2))]);
            Ptheta_az_k0 =SIG_THETA^2*diag([1;(2/(delta^2))]);
            Ptheta_el_k0 =SIG_THETA^2*diag([1;(2/(delta^2))]);

        elseif guidance_law == 2 || guidance_law == 3 % DG OR APN
        Prng_k0 = SIG_RNG^2*diag([1;(2/(delta^2));(4/(delta^4))]);
        Ptheta_az_k0
=SIG_THETA^2*diag([1;(2/(delta^2));(4/(delta^4))]);
        Ptheta_el_k0
=SIG_THETA^2*diag([1;(2/(delta^2));(4/(delta^4))]);
        end
    end
    %FINDING MEASUREMENT ESTIMATES
if kk >= 4

    if guidance_law == 1 %PN filters
        [range_k1, rangedot_k1, Prng ] = kalman_pn_range(range_k0,...
            rangedot_k0, nz_range, Prng_k0, mac_paraL_k1);
        [theta_az_k1, theta_az_dot_k1, Ptheta_az ] = kalman_pn_theta...
            (theta_az_k0, theta_az_dot_k0, nz_theta_az, Ptheta_az_k0,
...
            mac_perpL1_k1, range_k1);
        [theta_el_k1, theta_el_dot_k1, Ptheta_el ] = kalman_pn_theta...
            (theta_el_k0, theta_el_dot_k0, nz_theta_el, Ptheta_el_k0,
...
            mac_perpL2_k1, range_k1);
    elseif guidance_law == 2 || guidance_law == 3 %DG filters
        [range_k1, rangedot_k1, rangedotdot_k1, Prng ] =
kalman_dg_range...
            (range_k0, rangedot_k0, rangedotdot_k0, nz_range, Prng_k0,
...
            mac_paraL_k1);
        [theta_az_k1, theta_az_dot_k1,...
          theta_az_dotdot_k1, Ptheta_az ] =
kalman_dg_theta(theta_az_k0,...
          theta_az_dot_k0,theta_az_dotdot_k0, nz_theta_az,
Ptheta_az_k0,...
            mac_perpL1_k1, range_k1);
        [theta_el_k1, theta_el_dot_k1,...
         theta_el_dotdot_k1, Ptheta_el ] =
kalman_dg_theta(theta_el_k0,...
         theta_el_dot_k0,theta_el_dotdot_k0, nz_theta_el, Ptheta_el_k0,
...
            mac_perpL2_k1, range_k1);
    end
end

    %FINDING ESTIMATES OF RANGE,VT,Vp
    if kk >= 4
        Range_k1=[  range_k1*cos(theta_el_k1)*cos(theta_az_k1);
                    range_k1*cos(theta_el_k1)*sin(theta_az_k1);
                    range_k1*sin(theta_el_k1)];
        Range_k1_out(:,kk)=Range_k1; % LOS vector estimate output
        Range_k1_u=Range_k1/range_k1; % unit vector of LOS estimate
        Vt_k1 = (Vm) +(Range_k1-Range_k1_out(:,kk-1) )/delta;
        Vr_k1=Vt_k1-Vm;% LOS velocity vector estimate
        VLperp=Vr_k1-dot(Vr_k1,Range_k1_u)*Range_k1_u;
        VLperp_u=VLperp/norm(VLperp);

        thetadot_k1=norm(VLperp)/range_k1; % estimate of LOS rate
        thetadot_k1_out(1,kk)=thetadot_k1; % output update
        thetaLM_k1 = dot(Range_k1_u,Vm_u); % missile lead angle
estimate
        %  target lead angle estimate
        thetaLT_k1=dot(Range_k1_u,Vt_k1/norm(Vt_k1));
    end
end
%% FINDING ESTIMATE OF TARGET ACCELERATION
if kk>4 && (guidance_law==2  || guidance_law == 3)
    if FILT == 0
        Acc_t_k1=(Vt-Vtout(:,kk-1))/delta;
    elseif FILT == 1
        % using dummy variables to build
        % estimate of range acceleration vector
        R=range_k1;
        Rdot=rangedot_k1;
        Rdot2=rangedotdot_k1+(mac_paraL_k1);
%         Update current time step corrected estimate
%         of range acceleration (rangedotdot_k1)to
%         include the deterministic acceleration input
%         (missile's drag and boost accelerations
%         perpendicular to the LOS). This term cannot
%         be added inside the filter because changes
%         in LOS angle acceleration are modeled by the
%         filter as white noise.
        a=theta_az_k1;
        adot=theta_az_dot_k1;
        adot2=theta_az_dotdot_k1 + mac_perpL1_k1/range_k1;
%         Update current time step corrected estimate
%         of LOS azimuth angle acceleration (theta_az_dotdot_k1)to
%         include the deterministic acceleration input
%         (missile's drag and boost accelerations
%         perpendicular to the LOS). This term cannot
%         be added inside the filter because changes
%         in LOS angle acceleration are modeled by the
%         filter as white noise.
        b=theta_el_k1;
        bdot=theta_el_dot_k1;
        bdot2=theta_el_dotdot_k1 + mac_perpL2_k1/range_k1;
%         Update current time step corrected estimate
%         of LOS elevation angle acceleration (theta_el_dotdot_k1)to
%         include the deterministic acceleration input
%         (missile's drag and boost accelerations
%         perpendicular to the LOS). This term cannot
%         be added inside the filter because changes
%         in LOS angle acceleration are modeled by the
%         filter as white noise.

        AA=-sin(a)*((adot2*R+2*adot*Rdot)*cos(b)-2*adot*bdot*R*sin(b))-
...
            cos(a)*((( adot^2+bdot^2)*R-Rdot2)*cos(b)+...
            (bdot2*R+2*bdot*Rdot)*sin(b));
        BB=cos(a)*((adot2*R+2*adot*Rdot)*cos(b)-2*adot*bdot*R*sin(b))-
...
            sin(a)*(((adot^2+bdot^2)*R-Rdot2)*cos(b)+....
            (bdot2*R+2*bdot*Rdot)*sin(b));
        CC=(bdot2*R+2*bdot*Rdot)*cos(b)+(-bdot^2*R+Rdot2)*sin(b);
        Rdotdot=[AA;BB;CC];
        Acc_t_k1 = Ma + Rdotdot;% estimate of range acceleration vector
        Acc_t_k1_out(:,kk)=Acc_t_k1;% output update
    end
end
%% MISSILE BOOST
    if kk<=burntime/delta
        mab=thrust/mass(time)/GRAV;% magnitude of thrust (g)
        Mburn=mab*GRAV*Vm/vm; % thrust acc vector (m/s)
        burn=1;% Maintains boost phase
        else
        % Terminates boost phase.
        burn=0;
        mab=0;
        Mburn=zeros(3,1);
    end
%%
% MISSILE GUIDANCE
if (kk>100)
    if FILT == 0 && rangedot<0
        % unit vector that shows direction of guidance acc vector
        nc=cross(Vm_u,cross(Vp_u,Vm_u));
        if guidance_law==1 % PN guidance
            % PN guidance acceleration (g).
            mag  = PN_GL(thetaLM, thetadot, rangedot);
            guidance_flag=1;
        elseif guidance_law==2 % DG guidance.
            % DG guidance acceleration (g).
            mag = DG_GL (Acc_t_k1,thetaLT,thetaLM,rangedot,thetadot);
            guidance_flag=1;
        elseif guidance_law==3
            % APN guidance acceleration (g).
             mag  = APN_GL (Acc_t_k1,thetaLM,rangedot,thetadot);
             guidance_flag=1;
        end
    end
    if FILT == 1 && rangedot_k1<0
        % unit vector that shows direction of guidance acc vector
        nc=cross(Vm_u,cross(VLperp_u,Vm_u));
        if guidance_law==1 % PN guidance
            % PN guidance acceleration (g).
            mag  = PN_GL(thetaLM_k1, thetadot_k1, rangedot_k1);
            guidance_flag=1;
        elseif guidance_law==2 % DG guidance.
            % DG guidance acceleration (g).
            mag = DG_GL(Acc_t_k1,thetaLT_k1 ,thetaLM_k1,...
                         rangedot_k1,thetadot_k1);
            guidance_flag=1;
        elseif guidance_law==3
            % APN guidance acceleration (g).
            mag = APN_GL (Acc_t_k1,thetaLM_k1,rangedot_k1,thetadot_k1);
            guidance_flag=1;
        end
%         Averages guidance output with three previous outputs.
%         The result is a much smoother guidance output.
        mag = (mag + mag_out(kk-1) + mag_out(kk-2)+ mag_out(kk-3))/4;
    end
else
    mag=0;
    nc=[0;0;0];
end

if abs(mag) > maxg
    mag = sign(mag)*maxg;% Reassigns guidance acceleration to the
                         % maximum value while still retaining the
sign.
end
% guidance acc vector in perpendicular to missile velocity
Mag=mag*nc*GRAV;

%% MISSILE DRAG
[ Fdp, mach,Cdp] = fdp_value(malt, vm, burn);% parasitic drag vector
Fdi = (abs(mag)/maxg) *4*Fdp;% induced drag vector

map = (Fdp )/(mass(time)*GRAV); % parasitic drag(g)
mai = (Fdi )/(mass(time)*GRAV); % induced drag (g)
mad = (Fdp + Fdi)/(mass(time)*GRAV);
Mad = mad*(-Vm_u)*GRAV; % total drag in(m/s^2) in opposite direction
                        % of missile velocity vector
mach_out=[mach_out,mach]; % updated the Mach number output
Cdp_out =[Cdp_out,Cdp];% updated the drag coeff output

%% total acc, divert ,tgo
Ma = Mag + Mad + Mburn-[0;0;GRAV];% total missile acceleration (m/s^2)
Divert  = divert (guidance_flag,mag,Divert ); % calculates divert
t_impact  = time_to_impact ( range, rangedot );% calculates time to
impact
%% POST-FILTERING
if FILT==1 && kk >= 4
    Mac = Mburn + Mad; % deterministic input acceleration
    mac_paraL_k1 = Mac'*Range_k1_u;
    Mac_paraL_k1 = mac_paraL_k1*Range_k1_u;
    % deterministic input acceleration used for range filter
    mac_paraL_k1 = -mac_paraL_k1;

    Mac_perpL_k1 = Mac - Mac_paraL_k1;
    sign_mac_perpL_k1 = 
sign(cross((Mac_perpL_k1/norm(Mac_perpL_k1)),...
                              Range_k1_u));
    if sign_mac_perpL_k1(3) < 0
        % determinicstic input acceleration used for LOS azimuth angle
        mac_perpL1_k1 = -norm( Mac_perpL_k1(1:2) );
    end
    % determinicstic input acceleration used for LOS elevation angle
    mac_perpL2_k1 = -Mac_perpL_k1(3);

    %  Update filter variables
    VM_old = Vm;
    thetaLM_k0 = thetaLM_k1;

    Ptheta_az_k0 = Ptheta_az;
    Ptheta_el_k0 = Ptheta_el;
    theta_az_k0 = theta_az_k1;
    theta_el_k0 = theta_el_k1;
    theta_az_dot_k0 = theta_az_dot_k1;
    theta_el_dot_k0 = theta_el_dot_k1;

    range_k0 = range_k1;
    rangedot_k0 = rangedot_k1;
    Prng_k0 = Prng;
    if guidance_law==2 || guidance_law == 3
        theta_az_dotdot_k0 = theta_az_dotdot_k1;
        theta_el_dotdot_k0 = theta_el_dotdot_k1;
        rangedotdot_k0 = rangedotdot_k1;
    end
end
%% OUTPUT MATRICES
    Range_out = [Range_out,Range];% LOS vector Output Matrix
    PoutM = [PoutM,Hp*M]; % Missile position output matrix
    PoutT = [PoutT,Hp*T]; % Target position output matrix
    Vm_out = [Vm_out,Vm];   % Missile Velocity Output Matrix
    vm_out = [vm_out,vm];   % Missile Speed
    Vtout=[Vtout,Vt];       % Target Velocity
    range_out = [range_out,range]; % range
    range_dot_out(1,kk) = rangedot;% range rate
    range_dotdot_out(1,kk) = rangedotdot;% range acceleration

    theta_az_out = [theta_az_out,theta_az];  % LOS azimuth
    theta_az_dot_out=[theta_az_dot_out,theta_az_dot];% LOS azimuth rate
    theta_az_dotdot_out(1,kk)=theta_az_dotdot;% LOS azimuth 
acceleration

    theta_el_out = [theta_el_out,theta_el]; % LOS elevation
    theta_el_dot_out=[theta_el_dot_out,theta_el_dot];% LOS elevation 
rate
    theta_el_dotdot_out(1,kk)=theta_el_dotdot;% LOS elevation 
acceleration

    acc_out = [acc_out,norm(Ma)/GRAV]; % total acc (g)
    mag_out=[mag_out, mag];  % guidance acc (g)
    mad_out=[mad_out, mad];  % drag (g)
    map_out = [map_out, map];% parasitic drag (g)
    mai_out = [mai_out, mai];
    thetadot_out=[thetadot_out thetadot]; % LOS rate
    mburn_out=[mburn_out, mab]; % thrust (g)
    v_c_out=[v_c_out,-rangedot]; % closing speed (m/s)
    %update time,simulation time,time to go
    time = time+delta;
    tout = [tout,time];
    t_impact_out=[t_impact_out,t_impact];

%% OUTPUT MATRICES FOR FILTERED MEASUREMENTS
if FILT==1 && kk >= 4
    % updates output matrices for range, range rate
    range_k1out(1,kk) = range_k1;
    rangedot_k1out(1,kk) = rangedot_k1;
    % updates output matrices for LOS azimuth, LOS azimuth rate
    theta_az_k1out(1,kk) = theta_az_k1;
    theta_az_dot_k1_out(1,kk) = theta_az_dot_k1;
    % updates output matrices for LOS elevation, LOS elevation rate
    theta_el_k1out(1,kk) = theta_el_k1;
    theta_el_dot_k1_out(1,kk) = theta_el_dot_k1;

    if guidance_law==2 || guidance_law == 3
        %updates acceleration output matrices for
        % range, LOS azimuth, LOS elevation
        rangedotdot_k1_out(1,kk) = rangedotdot_k1+(mac_paraL_k1);
        theta_az_dotdot_k1_out(1,kk) =theta_az_dotdot_k1 +...
                                        mac_perpL1_k1/range_k1;
        theta_el_dotdot_k1_out(1,kk) = theta_el_dotdot_k1 +...
                                        mac_perpL2_k1/range_k1;
    end
end
%% STOP the simulation
% stops the simulation when closing speed turns negative(range rate
turn
% into positive)AND after guidance law implementation starts AND
missile
% speed is higher than target speed. Also simulation will run until 3.5
% seconds, which is less than minimum estimated simulation time.

% simulation stops when the minimum range (miss distance gets below 5
% meters. Simulation will also stop if the missile's altitude goes
below
% grounds level
if (rangedot>0 && guidance_flag==1 && (vm>vt)&& time>3.5  )...
        || (range<5)  || malt>0
    break
end
%% %MISSILE MOTION
    M=missile_motion(Ma,M );    % Ma is in m's^2
% TARGET MOTION
    T = target_motion(T,TURN);
```

```
 end
 %% Output of the results
RES=[sum(Divert(1,:));...% divert
    vm;...                 % impact velocity
    max(vm_out);...        % max missile speed
    range;...              % miss distance
    time;];                % impact time
```

```
format long;  close all; clear all;
% NFmax
%-----------------------------------------------------------------------
%-----------------------------------------------------------------------
%   File:                NFmax.m
%   Name:                1st LT Murat DOGEN
%   Component Runtime:   8.5.0.197613 (R2015a)
%   Compiler:            6 (R2015a)
%                        64-bit (Windows 8.1)
%   Date:                06 June 2015
%   Description:         Finds the
%-----------------------------------------------------------------------
% Set simulation parameters
TURN = 1; % maneuvering target
NZ = 1; % NOISE ON
NZ_FACTOR = 1; % starting from 1
FILT = 1;      % Filtering on
ANG = 0:1:180;  % aspect angles
RUNS = 50;      % number of iterations/runs
init ;         % installing global parameters
%select target distance (km)
D_target = 2;
% D_target = 8;
% D_target = 15;
guidance_law = 1; % select the GL to examine
fstep=[10,1,0.1]; % noise factor increment step size

max_factor = zeros(length(ANG),1);

for aa=1:length(ANG)    % One cycle for each aspect angle.
Ti = [D_target*1000 tspeed*cosd(ANG(aa)) 0 tspeed*sind(ANG(aa)) -alt_T
0]';
noise_factor=0;
    %% First test loop (step size = 10)
 for ii=1:length(fstep)
     for NZ_FACTOR = noise_factor+fstep(ii) : fstep(ii) :
noise_factor+9*fstep(ii)
        disp(['*** Noise Factor =',num2str(NZ_FACTOR),...
            ', step size = ',num2str(fstep(ii)),' ***  '])
        %Reset variables
        misses = 0;
        hits = 0;
        swings = 0;
        % Update sensor variance
        SIG_THETA = NZ_FACTOR*0.001;    % LOS angle sensor uncertainty
        SIG_RNG = NZ_FACTOR*10;         % Range sensor uncertainty (m)
        % Determine if missile is effective at this noise level
        while swings <= RUNS
            B0;     % Run simulation
            % Determine if missile hit
        if RES(4)>= 5
         disp( [' angle ' num2str(ANG(aa)),...
                    ' !!!! MISS !!!! ', num2str(RES(4)),...
                     ' MissileSpeed=  ',num2str(vm)])
```

```
             misses = misses + 1;
         else
          disp( [' angle ' num2str(ANG(aa)),...
                     ' hit  ', num2str(RES(4)),...
                      ' MissileSpeed=  ',num2str(vm)])
               hits = hits + 1;
         end
             % Missile is ineffective if it misses 30% of the time
             if misses == 0.3*RUNS+1;
                 noise_factor = NZ_FACTOR-fstep(ii);
                 break
             end
             % Missile is effective if it hits 70% of the time
             if hits == 0.7*RUNS;
                 noise_factor = NZ_FACTOR;
                 break
             end
             swings = swings + 1; % Increment count of simulations
         end
         % If missile is ineffective, move to the next test loop
         if misses == 0.3*RUNS+1;
             break
         end
     end
      %% Second test loop (step size = 1)
   if ii~=3
      for NZ_FACTOR = (noise_factor+fstep(ii)/2)
         disp(['*** Noise Factor =',num2str(NZ_FACTOR),...
             ', step size = ',num2str(fstep(ii)),' ***  '])
         %Reset variables
         misses = 0;
         hits = 0;
         swings = 0;
         % Update sensor variance
         SIG_THETA = NZ_FACTOR*0.001;    % LOS angle sensor uncertainty
         SIG_RNG = NZ_FACTOR*10;         % Range sensor uncertainty (m)

         % Determine if missile is effective at this noise level
         while swings <= RUNS
         B0;    % run the simulation script
             % Determine if missile hit
         if RES(4)>= 5
          disp( [' angle ' num2str(ANG(aa)),...
                     ' !!!! MISS !!!! ', num2str(RES(4)),...
                      ' MissileSpeed=  ',num2str(vm)])
             misses = misses + 1;
         else
          disp( [' angle ' num2str(ANG(aa)),...
                     ' hit  ', num2str(RES(4)),...
                      ' MissileSpeed=  ',num2str(vm)])
               hits = hits + 1;
         end
             % Missile is ineffective if it misses 30% of the time
             if misses == 0.3*RUNS+1;
```

```
                noise_factor = NZ_FACTOR-fstep(ii)/2;
                break
            end
            % Missile is effective if it hits 70% of the time
            if hits == 0.7*RUNS;
                noise_factor = NZ_FACTOR;
                break
            end
            swings = swings + 1; % Increment count of simulations
        end
        % If missile is ineffective, move to the next test loop
        if misses == 0.3*RUNS+1;
            break
        end
     end
 end
 end
    % Update max_factor plotting vector
    max_factor(aa) = noise_factor;
end
```

B. SIMULATION GUIDANCE LAW FILES

```
function [ mag ] = PN_GL(thetaLM, thetadot, rangedot)
% Computes the PN guidance acceleration magnitude
%-----------------------------------------------------------------------
%   File:               PN_GL.m
%   Name:               1st LT Murat DOGEN
%   Component Runtime:  8.5.0.197613 (R2015a)
%   Compiler:           6 (R2015a)
%                       64-bit (Windows 8.1)
%   Date:               06 June 2015
%   Description:        Uses the proportional navigation guidance law
%                       to compute the magnitude of the missile.
%                       guidance acceleration vector (mag).
%                       Allows user to modify initialization parameters
%   Inputs:             Missile lead angle (thetaLM) (rad).
%                       LOS angle rate (thetadot) (rad/sec).
%                       range rate (rangedot) (m/sec).
%   Outputs:            Magnitude of the missile guidance acceleration
%                       vector (mag) (g).
----------------------------------------------------------------------
%------ define globals ------
global Nprime GRAV
% Magnitude of missile guidance acceleration vector (g)
mag=Nprime*(-rangedot)*thetadot/thetaLM/GRAV;
end
```

```
function [ mag ] = DG_GL (Acc_t_k1,thetaLT,thetaLM,rangedot,thetadot)
% Computes the DG guidance acceleration magnitude
%----------------------------------------------------------------------
%   File:               DG_GL.m
%   Name:               1st LT Murat DOGEN
%   Component Runtime:  8.5.0.197613 (R2015a)
%   Compiler:           6 (R2015a)
%                       64-bit (Windows 8.1)
%   Date:               06 June 2015
%   Description:        Uses the differential geometry guidance law
%                       to compute the magnitude of the missile
%                       guidance acceleration vector (mag).
%                       Allows user to modify initialization parameters
%   Inputs:             Estimate of target acceleration vector
(Acc_t_k1)
%                       Missile lead angle (thetaLM) (rad)
%                       Target lead angle (thetaLT) (rad)
%                       LOS angle rate (thetadot) (rad/sec)
%                       range rate (rangedot) (m/sec)
%   Outputs:            Magnitude of the missile guidance acceleration
%                       vector (mag) (g).
%----------------------------------------------------------------------
%------ define globals ------
global Nprime GRAV
% Magnitude of missile guidance acceleration vector (g)
mag=norm(Acc_t_k1)*thetaLT/thetaLM/GRAV + ...
    (Nprime*(-rangedot)*thetadot)/thetaLM/GRAV ;
end
```

```
function [ mag ] = APN_GL (Acc_t_k1,thetaLM,rangedot,thetadot)
% Computes the APN guidance acceleration magnitude
%-----------------------------------------------------------------------
%   File:                APN_GL.m
%   Name:                1st LT Murat DOGEN
%   Component Runtime:   8.5.0.197613 (R2015a)
%   Compiler:            6 (R2015a)
%                        64-bit (Windows 8.1)
%   Date:                06 June 2015
%   Description:         Uses the augmented proportional navigation
%                        guidance law compute the magnitude of the
%                        missile guidance acceleration vector (mag).
%   Inputs:              Estimate of target acceleration vector
%                        (Acc_t_k1).
%                        Missile lead angle (thetaLM) (rad).
%                        LOS angle rate (thetadot) (rad/sec).
%                        range rate (rangedot) (m/sec).
%   Outputs:             Magnitude of the missile guidance acceleration
%                        vector (mag) (g).
%-----------------------------------------------------------------------
%------ define globals ------
global Nprime GRAV
% Calculate the magnitude of the missile guidance acceleration vector.
mag=( Nprime*(-rangedot)*thetadot +...
    0.5*Nprime*norm(Acc_t_k1) )/thetaLM/GRAV;
end
```

C. SIMULATION FUNCTION FILES

```
function [ tgo ] = time_to_impact ( range, rangedot )
% TIME_TO_IMPACT
% Computes time to impact with target.
%-----------------------------------------------------------------------
%-----------------------------------------------------------------------
%   File:                time_to_impact.m
%   Name:                1st LT Murat DOGEN
%   Component Runtime:   8.5.0.197613 (R2015a)
%   Compiler:            6 (R2015a)
%                        64-bit (Windows 8.1)
%   Date:                06 June 2015
%   Description:         Computes time remaining until impact (tgo) with
%                        target based on range and range rate.
%   Inputs:              LOS range (range) (m).
%                        LOS range rate (rangedot) (m/sec)
%   Outputs:             Time remaining until impact (tgo) (sec).
%   Comments:            Portions of this code have been reused from
%                        Osborn's Thesis.
%-----------------------------------------------------------------------
if (rangedot == 0) % Prevent dividing by zero when rangedot = 0.
    tgo = 15; % Choose some large number to prevent turn at the
              % beginning of the simulation.
else
    tgo = range / (-rangedot);
end
end
```

```
function [ mach_speed ] = mach_speed ( malt )
% MACH_SPEED
% Computes Mach speed (m/sec) for a given altitude.
%-----------------------------------------------------------------------
%   File:                 mach_speed.m
%   Name:                 1st LT Murat DOGEN
%   Component Runtime:    8.5.0.197613 (R2015a)
%   Compiler:             6 (R2015a)
%                         64-bit (Windows 8.1)
%   Date:                 06 June 2015
%   Description:          Computes the linear approximation of Mach 1
%                         velocity (m/sec) for a given altitude based on
%                         standard ICAO atmosphere.
%   Inputs:               Missile altitude (malt) (m)
%   Outputs:              Velocity of Mach 1 (mach_speed) (m/sec)
%   Process:              Uses polynomial fit of the altitude/velocity
%                         curve of Mach 1 in a standard ICAO atmosphere.
%   Comments:             Portions of this code have been reused from
%                         Osborn's Thesis.
%------ define constants ------
% Constants for altitude/velocity curve in standard ICAO atmosphere.
A = [-0.0041 340.3]; % Altitudes below 11km.
B = 295.1;           % Altitude of 11-20km.
C = [0.00067 281.7]; % Altitudes greater than 20km.
%------ define input vector ------
%------ initialize variables ------
malt = abs(malt); % Absolute value accounts for NED coordinate system
%------ functions ------
if (malt<11000) % Use A variables if altitude is below 11km.
    mach_speed = polyval(A,malt); % Velocity of Mach 1 (m/sec).
elseif (malt>20000) % Use B variables if altitude is above 20km.
    mach_speed = polyval(C,malt); % Velocity of Mach 1 (m/sec).
else
    mach_speed = B; % Velocity of Mach 1 (m/sec).
end
end
```

```
function [ MASS ] = mass (t )
% MASS
% Computes Mach speed (m/sec) for a given altitude.
%-----------------------------------------------------------------------
%   File:                mass.m
%   Name:                1st LT Murat DOGEN
%   Component Runtime:  8.5.0.197613 (R2015a)
%   Compiler:            6 (R2015a)
%                        64-bit (Windows 8.1)
%   Date:                06 June 2015
%   Description:         Computes the mass of the missile at the
simulation
%                        time
%   Inputs:              Simulation time (t) (sec)
%   Outputs:             Missile Mass (MASS) (kg)
%   Process:             As the propellant burns, the mass of the
missile
%                        will decrease to impact mass. This decrease
happens
%                        in burn-time duration.
%-----------------------------------------------------------------------
%------ define globals ------
global burntime mass_total mass_impact
if t <= burntime
    % Missile mass in burn phase
    MASS=-(mass_total-mass_impact)*t/burntime+mass_total;
else
    % Missile mass in after burnout
    MASS=mass_impact;
end
end
```

```matlab
function [ rho ] = rho_value ( malt )
% RHO_VALUE
% Computes the atmospheric density.
%----------------------------------------------------------------------
%   File:                rho_value.m
%   Name:                1st LT Murat DOGEN
%   Component Runtime:   8.5.0.197613 (R2015a)
%   Compiler:            6 (R2015a)
%                        64-bit (Windows 8.1)
%   Date:                06 June 2015
%   Description:         Computes the atmospheric density at the given
%                        altitude for ICAO standard atmosphere.
%   Inputs:              Missile altitude (malt) (m)
%   Outputs:             Atmospheric density (rho) (kg/m^3)
%   Comments:            Portions of this code have been reused from
%                        Pehr’s Thesis.
%----------------------------------------------------------------------
%------ initialize variables ------
malt = abs(malt); % Absolute value accounts for NED coordinates.
%------ functions ------
if malt > 9144
    % Atmospheric density below 9144 meters.
    rho = 1.75228763*exp(-malt/6705.6);
else
    % Atmospheric density above or at 9144 meters.
    rho = 1.22557*exp(-malt/9144);
end
end
```

```
function [Cdp ] = cdp_value (Mach, burn)
% CDP_VALUE
% Computes approximation of zero lift drag coefficient.
%----------------------------------------------------------------------
%   File:                 cdp_value.m
%   Name:                 1st LT Murat DOGEN
%   Component Runtime:    8.5.0.197613 (R2015a)
%   Compiler:             6 (R2015a)
%                         64-bit (Windows 8.1)
%   Date:                 06 June 2015
%   Description:          Computes approximation of zero lift drag
%                         coefficient from graph of Cdp vs. Mach number.
%   Inputs:               Missile Mach number (mach) (unitless)
%                         Burn value (boost) (1 during "boost phase", 0
%                         otherwise)
%   Outputs:              Parasitic drag coefficient (Cdp) (unitless)
%   Process:              Uses polynomial fit of the parasitic drag
%                         coefficient curve given in Pehr thesis pg. 18.
%   Comments:             Portions of this code have been reused from
%                         Osborn’s Thesis and his parasitic model has
been
%                         modified to get one similar to the empirical
drag
%                         model
%----------------------------------------------------------------------
% Constants for the parasitic drag coefficient curve
Mb=[1.2  1.7 2    2.5   3      3.5   4    5    6 ];
Db=[0.27 0.2 0.18 0.15   0.135  0.12  0.11 0.1  0.095];
Boost=polyfit(Mb,Db,4);

Mnb=[1.2  1.4 2    2.5    3     3.4   4    5        6 ];
Dnb=[0.46 0.4 0.29 0.25   0.22  0.2  0.18 0.155  0.132];
NoBoost=polyfit(Mnb,Dnb,4);

if (burn & (mach<0.8))
    Cdp=0.15;
end
if (~burn & (mach<0.8))
    Cdp=0.25;
end
if (burn & (mach>=0.8) & (mach<1.2))
    Cdp=(mach-0.8)*0.24 + 0.15;
end
if (~burn & (mach>=0.8) & (mach<1.2))
    Cdp=(mach-0.8)*0.42 + 0.25;
end
if ((mach>=1.2) & (burn~=0))
    Cdp=polyval(Boost, mach);
end
if ((mach>=1.2) & (burn==0))
    Cdp=polyval(NoBoost, mach);
end
end
```

```
function [ Fdp, mach,Cdp] = fdp_value (malt, mspd, burn)
% FDP_VALUE
% Computes missile's parasitic drag force.
%-----------------------------------------------------------------------
%   File:               fdp_value.m
%   Name:               1st LT Murat DOGEN
%   Component Runtime:  8.5.0.197613 (R2015a)
%   Compiler:           6 (R2015a)
%                       64-bit (Windows 8.1)
%   Date:               06 June 2015
%   Description:        Computes the parasitic drag force for a missile
%                       with frontal area SREF in standard atmosphere.
%   Inputs:             Missile altitude (malt) (m)
%                       Missile speed (mspd) (m/sec)
%                       Boost value (burn) (1 during "boost phase", 0
%                       otherwise)
%   Outputs:            Parasitic drag force (Fdp) (N)
%   Comments:           Portions of this code have been reused from
%                       Osborn's Thesis.
%-----------------------------------------------------------------------
%------ define globals ------
global SREF
%------ initialize variables ------
rho = rho_value(malt);        % Atmospheric density (kg/m^3).
mach = mspd/mach_speed(malt); % Missile speed as Mach number
Q = rho*mspd^2/2;             % Dynamic pressure (kg*m/sec^2 or N)
Cdp = cdp_value(mach,burn);   % Parasitic drag coefficient(unitless).
%------ functions ------
%% Determine parasitic drag force
Fdp = Cdp*Q*SREF; % Force due to parasitic drag (kg*m/s^2 or N)
end
```

```
function [nz_range,  nz_theta_az,  nz_theta_el ] =
noisy_data(range,theta_az,theta_el)
% NOISY_RANGE
% Adds noise to range measurements.
%----------------------------------------------------------------------
%   File:                noisy_data.m
%   Name:                1st LT Murat DOGEN
%   Component Runtime:   8.5.0.197613 (R2015a)
%   Compiler:            6 (R2015a)
%                        64-bit (Windows 8.1)
%   Date:                06 June 2015
%   Description:         Adds white noise to range based on noise
%                        multiplier of randn.
%   Inputs:              Actual LOS range (range) (m)
%                        Actual LOS azimuth angle (theta_az) (rad)
%                        Actual LOS elevation angle (theta_el) (rad)
%   Outputs:             Noisy LOS range measurement (nz_range) (m)
%                        Noisy LOS azimuth angle measurement
(nz_theta_az) (m)
%                        Noisy LOS elevation angle measurement
(nz_theta_el) (m)
%   Process:             Global variable NZ_FACTOR is used as a
multiplier
%                        to the sensor accuracy (SIG_RNG) and
(SIG_THETA).
%                        The noisy measurements are generated by
%                        adding white noise with the randn function.
%----------------------------------------------------------------------
%------ define globals ------
global SIG_RNG SIG_THETA
% Generate  noisy measurements.
nz_range    = range     + randn * SIG_RNG;
nz_theta_az = theta_az  + randn * SIG_THETA;
nz_theta_el = theta_el  + randn * SIG_THETA;
end
```

```
function [ M ] = missile_motion(Ma,M )
% MISSILE_MOTION
% Updates missile state vector (M).
%----------------------------------------------------------------------
%   File:                   missile_motion.m
%   Name:                   1st LT Murat DOGEN
%   Component Runtime:      8.5.0.197613 (R2015a)
%   Compiler:               6 (R2015a)
%                           64-bit (Windows 8.1)
%   Date:                   06 June 2015
%   Description:            Computes the updated missile state vector (M)
%                           including straight line motion and changes due
%                           to total acceleration (drag, guidance, boost).
%   Inputs:                 Missile state vector (M)
%                           Missile total acceleration (Ma)
%   Outputs:                Updated Missile state vector (M)
%   Process:                Adds straight line motion effects with the
%                           accelerations due to drag, guidance, and
thrust.
%   Comments:               Portions of this code have been reused from
%                           Osborn's Thesis.
%----------------------------------------------------------------------
%------ define globals ------
global delta
Fm = [0 1 0 0 0 0;0 0 0 0 0 0;
      0 0 0 1 0 0;0 0 0 0 0 0;
      0 0 0 0 0 1;0 0 0 0 0 0];
% Missile transition matrix for straight line motion.
Fm = expm(Fm*delta);
%Missile Motion
    xmzz = Ma*delta^2/2; %change in missile position with const acc
    vmzz = Ma*delta;     %change in missile velocity with const acc
% Total change in missile state other than straight line motion
Xmz = [xmzz(1),vmzz(1),xmzz(2),vmzz(2),xmzz(3),vmzz(3)]';
%Updated missile state vector
M = Fm*M + Xmz;   % Adds straight line motion effects with the
                  % accelerations due to drag, guidance, and thrust.
end
```

```
function [ T ] = target_motion(T,TURN)
% TARGET_MOTION
% Updates target state vector (T).
%-----------------------------------------------------------------------
%-----------------------------------------------------------------------
%   File:                target_motion.m
%   Name:                1st LT Murat DOGEN
%   Component Runtime:   8.5.0.197613 (R2015a)
%   Compiler:            6 (R2015a)
%                        64-bit (Windows 8.1)
%   Date:                06 June 2015
%   Description:         Computes the updated target state vector (T)
%                        during straight line motion and changes due to
%                        turn acceleration. Turn is based on turn value.
%                        Target conducts turn toward missile of
%                        magnitude TargetTurnG (g) when turn value is 1.
%                        This occurs at 3 seconds before the impact.
%   Inputs:              Target state vector (T)
%                        TURN value (0 for no turn, 1 for turn)
%   Outputs:             Updated target state vector (T)
%   Process:             The straight line motion transition matrix is
%                        used when TURN value is 0. The turn transition
%                        matrix is used when TURN value is 1.
%-----------------------------------------------------------------------
%------ define globals ------
global TargetTurnG delta tspeed t_impact GRAV
% Angular acceleration of Target
w = TargetTurnG*GRAV/tspeed;
% Motion
%     if abs(t_impact)<=3.0
%         turn=1;
%     else
%         turn=0;
%     end
if TURN==1 % simulation parameter TURN is 1, then check if the impact
           % time is less than 3 seconds to initiate the target turn
    if abs(t_impact)<=3.0
        Fs = [0 1 0 0 0 0;0 0 0 -w 0 0;
            0 0 0 1 0 0;0 w 0  0 0 0;
            0 0 0 0 0 1;0 0 0  0 0 0];
        % Target transition matrix during a turn.
        Fturn=expm(Fs*delta);
        Fs=Fturn;
    end
else
    Fs = [0 1 0 0 0 0;0 0 0 0 0 0;
        0 0 0 1 0 0;0 0 0 0 0 0;
        0 0 0 0 0 1;0 0 0 0 0 0];
        % Target transition matrix for straight line motion.
        Fs = expm(Fs*delta);
end
% Updated target state matrix after target motion
T = Fs*T;
end
```

```
function [ Divert ] = divert (  GF,ma,Divert )
% DIVERT
% Generates global variables in the workspace.
%----------------------------------------------------------------------
%   File:               divert.m
%   Name:               1st LT Murat DOGEN
%   Component Runtime:  8.5.0.197613 (R2015a)
%   Compiler:           6 (R2015a)
%                       64-bit (Windows 8.1)
%   Date:               06 June 2015
%   Description:        Sums the magnitude of the guidance acceleration
%                       command after guidance flag turns to 1.
%----------------------------------------------------------------------
if GF == 1
    D1 =norm(ma);
    Divert=[Divert,[D1;GF]];
elseif GF == 0
    Divert=[Divert,[0;GF]];
end
end
```

D. SIMULATION FILTER FILES

```
function [rng_k1, rngdt_k1, Prng_k1] = kalman_pn_range...
    (range_k0, rangedot_k0, nz_range, Prng_k0, mac_paraL_k1)
% KALMAN_PN_RANGE
%-----------------------------------------------------------------------
%   File:                kalman_pn_range.m
%   Name:                1st LT Murat DOGEN
%   Component Runtime:   8.5.0.197613 (R2015a)
%   Compiler:            6 (R2015a)
%                        64-bit (Windows 8.1)
%   Date:                06 June 2015
%   Description:         Kalman algorithm which generates current
%                        discrete time step corrected estimates of range
%                        and range rate as well as the corrected
%                        covariance of the estimation.
%   Inputs:              Kalman filter's previous time step estimate of:
%                        range (rng_k0) (m)
%                        range rate (rngdt_k0) (m/sec)
%                        Kalman filter's previous time step corrected
%                        range covariance (Prng_k0)
%                        Noisy range measurement (nz_rng) (m)
%                        Estimated magnitude of combined deterministic
%                        missile acceleration parallel to the LOS
%                        (mac_paraL_k1).
%   Outputs:             Kalman filter's current time step estimate of:
%                        range (rng_k1) (m)
%                        range rate (rngdt_k1) (m/sec)
%                        Kalman filter's current time step corrected
%                        range covariance (Prng_k1)
%-----------------------------------------------------------------------
global delta SIG_RNG  GRAV TargetTurnG
%------ define constants ------
% Transition matrix
F = [1, delta;
    0, 1];
% Covariance matrix for uncorrelated bearing measurements
R = (SIG_RNG^2);
% Measurement matrix
H =[1, 0];
% Plant noise gain matrix
G = eye(2);
%------ define input vector ------
% Set corrected theta state values from previous discrete time step
xc = [ range_k0;
       rangedot_k0];
Pc = Prng_k0;
%------ initialize variables ------
% Plant noise covariance multiplier
q2 = 3*((TargetTurnG*GRAV)^2)*delta/4;
% Plant noise covariance
Q = q2*[(delta^3)/3, (delta^2)/2;
        (delta^2)/2, delta];
%------ functions ------
```

```
%% Prediction phase
% Predicted range state
u = (mac_paraL_k1)*[(delta^2)/2; delta];
xp = F*xc + G*u; % Predicted range state
Pp = F*Pc*F' + Q;% Predicted range state estimation covariance
%% Correction Phase
S = R + H*Pp*H'; % Innovation covariance
W = S\(Pp*H'); %Kalman Gain (W)
v = nz_range - H*xp;% innovations
xc = xp + W*v;% corrected state vector (xc)
Pc = (eye(2)-W*H)*Pp*(eye(2)-W*H)' + W*R*W';%corrected covariance (Pc)
% Corrected range state values for current discrete time step
rng_k1 = xc(1);
rngdt_k1 = xc(2);
Prng_k1 = Pc;
end
```

```
function [theta_k1, thetadt_k1, Ptheta_k1] = kalman_pn_theta...
(theta_k0, thetadot_k0, nz_theta, Ptheta_k0, mac_perpL_k1, range_k1)
% KALMAN_PN_THETA
%-----------------------------------------------------------------------
%   File:                kalman_pn_theta.m
%   Name:                1st LT Murat DOGEN
%   Component Runtime:   8.5.0.197613 (R2015a)
%   Compiler:            6 (R2015a)
%                        64-bit (Windows 8.1)
%   Date:                06 June 2015
%   Description:         Kalman algorithm which generates current
%                        discrete time step corrected estimates of theta
%                        and theta rate as well as the corrected
%                        covariance of the estimation.
%   Inputs:              Kalman filter's previous time step estimate of:
%                        LOS angle (theta_k0) (rad)
%                        LOS angle rate (thetadot_k0) (rad/sec)
%                        Kalman filter's previous time step corrected
%                        theta covariance (Ptheta_k0)
%                        Kalman filter's current time step estimate of
%                        range (rng_k1) (m)
%                        Noisy theta measurement (nz_theta) (rad)
%                        Estimated magnitude of combined deterministic
%                        missile acceleration perpendicular to the LOS
%                        (mac_perpL_k1).
%   Outputs:             Kalman filter's current time step estimate of:
%                        LOS angle (theta_k1) (rad)
%                        LOS angle rate(thetadot_k1) (rad/sec)
%                        Kalman filter's current time step corrected
%                        theta covariance (Ptheta_k1)
%   Comments:            mac_perpL1_k1 is used for LOS azimuth angle
%                        mac_perpL1_k2 is used for LOS elevation angle
%-----------------------------------------------------------------------
%------ define globals ------
global delta SIG_THETA GRAV TargetTurnG
%------ define constants ------
% Transition matrix
F = [1, delta;
     0, 1];
% Covariance matrix for uncorrelated bearing measurements
R = (SIG_THETA^2);
% Measurement matrix
H =[1, 0];
% Plant noise gain matrix
G = eye(2);
%------ define input vector ------
% Set corrected theta state values from previous discrete time step
xc = [  theta_k0;
        thetadot_k0];
Pc = Ptheta_k0;
%------ initialize variables ------
% Plant noise covariance multiplier
q2 = 3*((TargetTurnG*GRAV)^2)*delta/(4*(range_k1^2));
% Plant noise covariance
```

```
Q = q2*[(delta^3)/3, (delta^2)/2;
        (delta^2)/2, delta];
%------ functions ------
%% Prediction phase
% Predicted theta state
u = [   atan2(mac_perpL_k1*(delta^2), 2*range_k1);
        4*mac_perpL_k1*range_k1*delta / ...
        (4*(range_k1^2) + (mac_perpL_k1)^2*(delta^4))];
xp = F*xc + G*u; % Predicted theta state
% Predicted range state estimation covariance
Pp = F*Pc*F' + Q;
%% Correction Phase
% Calculate Kalman Gain (W)
S = R + H*Pp*H'; % Innovation covariance
W = S\(Pp*H'); %  Kalman Gain (W)
v = nz_theta - H*xp;%  the innovations
xc = xp + W*v;%  corrected state vector (xc)
Pc = (eye(2)-W*H)*Pp*(eye(2)-W*H)' + W*R*W';% corrected covariance (Pc)
% Corrected theta state values for current discrete time step
theta_k1 = xc(1);
thetadt_k1 = xc(2);
Ptheta_k1 = Pc;
end
```

```
function [range_k1, rangedot_k1, rangedotdot_k1, Prng_k1] =
kalman_dg_range...
    (range_k0, rangedot_k0, rangedotdot_k0, nz_range, Prng_k0,
mac_paraL_k1)
% KALMAN_DG_RANGE
%----------------------------------------------------------------------
%   File:                 kalman_dg_range.m
%   Name:                 1st LT Murat DOGEN
%   Component Runtime:    8.5.0.197613 (R2015a)
%   Compiler:             6 (R2015a)
%                         64-bit (Windows 8.1)
%   Date:                 06 June 2015
%   Description:          Kalman algorithm which generates current
%                         discrete time step corrected estimates of
%                         range, range rate, and range acceleration as
%                         well as the corrected covariance of the
%                         estimation.
%   Inputs:               Kalman filter's previous time step estimate of:
%                         range (range_k0) (m)
%                         range rate (rangedot_k0) (m/sec)
%                         range acceleration (rangedotdot_k0) (m/sec^2)
%                         Kalman filter's previous time step corrected
%                         range covariance (Prng_k0)
%                         Noisy range measurement (nz_range) (m)
%                         Estimated magnitude of combined deterministic
%                         missile acceleration parallel to the LOS
%                         (mac_paraL_k1).
%   Outputs:              Kalman filter's current time step estimate of:
%                         range (range_k1) (m)
%                         range rate (rangedot_k1) (m/sec)
%                         range acceleration (rangedotdot_k1) (m/sec^2)
%                         Kalman filter's current time step corrected
%                         range covariance (Prng_k1)
%   Comments:             Position and velocity portions of the missile's
%                         deterministic acceleration inputs are applied
%                         inside the filter. The acceleration portion is
%                         applied outside the filter since unpredictable
%                         accelerations are modeled as white noise inside
%                         the filter.
%----------------------------------------------------------------------
%----------------------------------------------------------------------
%------ define globals ------
global delta SIG_RNG  GRAV TargetTurnG
%------ define constants ------
% Transition matrix
F = [1,delta,(delta^2)/2;
    0, 1, delta;
    0, 0, 1];
% Covariance matrix for uncorrelated range measurements
R = (SIG_RNG^2);
% Measurement matrix
H =[1, 0, 0];
% Plant noise gain matrix
G = eye(3);
```

```
%------ define input vector ------
% Set corrected range state values from previous discrete time step
xc = [  range_k0;
        rangedot_k0;
        rangedotdot_k0];
Pc = Prng_k0;
%------ initialize variables ------
% Plant noise covariance multiplier
q2 = 3*delta*((TargetTurnG*GRAV)^2)/4;

Q = q2*[(delta^5)/20, (delta^4)/8, (delta^3)/6;
        (delta^4)/8, (delta^3)/3, (delta^2)/2;
        (delta^3)/6, (delta^2)/2, delta];
%------ functions ------

%% Prediction phase
u = (mac_paraL_k1)*[(delta^2)/2; delta; 0];% Predicted range state
xp = F*xc + G*u; % Predicted range state
Pp = F*Pc*F' + Q;% Predicted range state estimation covariance

%% Correction Phase
S = R + H*Pp*H';    % Innovation covariance
W = S\(Pp*H');      % Kalman Gain (W)
z = nz_range - H*xp;%  the innovations
xc = xp + W*z;%  corrected state vector (xc)
Pc = (eye(3)-W*H)*Pp*(eye(3)-W*H)' + W*R*W'; % corrected covariance
% Corrected range state values for current discrete time step
range_k1 = xc(1);
rangedot_k1 = xc(2);
rangedotdot_k1 = xc(3);
Prng_k1 = Pc;
end
```

```
function [theta_k1, thetadot_k1, thetadotdot_k1, Ptheta_k1] = ...
    kalman_dg_theta(theta_k0, thetadot_k0, thetadotdot_k0, nz_theta,
...
    Ptheta_k0, mac_perpL_k1, range_k1)
% KALMAN_DG_THETA
%-----------------------------------------------------------------------
%   File:                 kalman_dg_theta.m
%   Name:                 1st LT Murat DOGEN
%   Component Runtime:    8.5.0.197613 (R2015a)
%   Compiler:             6 (R2015a)
%                         64-bit (Windows 8.1)
%   Date:                 06 June 2015
%   Description:          Kalman algorithm which generates current
%                         discrete time step corrected estimates of LOS
%                         angles, its first and second time derivatives
and
%                         well as the corrected covariance.
%   Inputs:               Kalman filter's previous time step estimate of:
%                         LOS angle (theta_k0) (rad)
%                         LOS angle rate (thetadot_k0) (rad/sec)
%                         LOS angle acceleration(thetadotdot_k0)
(rad/sec^2)
%                         Kalman filter's previous time step corrected
%                         theta covariance (Ptheta_k0)
%                         Kalman filter's current time step estimate of
%                         range (range_k1) (m)
%                         Noisy theta measurement (nz_theta) (rad)
%                         Estimated magnitude of combined deterministic
%                         missile acceleration perpendicular to the LOS
%                         (mac_perpL_k1).
%   Outputs:              Kalman filter's current time step estimate of:
%                         LOS angle (theta_k1) (rad)
%                         LOS angle rate(thetadot_k1) (rad/sec)
%                         LOS angle acceleration(thetadotdot_k1)
(rad/sec^2)
%                         Kalman filter's current time step corrected
%                         theta covariance (Ptheta_k1)
%   Comments:             mac_perpL1_k1 is used for LOS azimuth angle
%                         mac_perpL1_k2 is used for LOS elevation angle
%-----------------------------------------------------------------------
%------ define globals ------
global delta SIG_THETA  GRAV TargetTurnG
%------ define constants ------
% Transition matrix
F = [1,delta,(delta^2)/2;
    0, 1, delta;
    0, 0, 1];
% Covariance matrix for uncorrelated bearing measurements
R = (SIG_THETA^2);
% Measurement matrix
H =[1, 0, 0];
% Plant noise gain matrix
G = eye(3);
%------ define input vector ------
```

```
% Set corrected theta state values from previous discrete time step
xc = [  theta_k0;
        thetadot_k0;
        thetadotdot_k0];
Pc = Ptheta_k0;
%------ initialize variables ------
% Plant noise covariance multiplier
q2 = 3*((TargetTurnG*GRAV)^2)*delta/(4*(range_k1^2));
if range_k1 >= 500
    q2 = 1*q2;
else
    q2 = 10*q2;
end
% Plant noise covariance
Q = q2*[(delta^5)/20, (delta^4)/8, (delta^3)/6;
        (delta^4)/8, (delta^3)/3, (delta^2)/2;
        (delta^3)/6, (delta^2)/2, delta];
%% Prediction phase
% Predicted theta state
u = [atan2(mac_perpL_k1*(delta^2), 2*range_k1);
     4*mac_perpL_k1*range_k1*delta / (4*(range_k1^2) +
(mac_perpL_k1)^2*(delta^4));
     0];
xp = F*xc + G*u; % Predicted theta state
Pp = F*Pc*F' + Q;% Predicted range state    estimation covariance
%% Correction Phase
S = R + H*Pp*H'; % Innovation covariance
W = S\(Pp*H');
v = nz_theta - H*xp; %innovations
% Calculate corrected state vector (xc)
xc = xp + W*v;
Pc = (eye(3)-W*H)*Pp*(eye(3)-W*H)' + W*R*W';% corrected covariance (Pc)
% Corrected theta state values for current discrete time step
theta_k1 = xc(1);
thetadot_k1 = xc(2);
thetadotdot_k1 = xc(3);
Ptheta_k1 = Pc;
end
```

LIST OF REFERENCES

[1] T. G. Mahnken, “The Cruise Missile Challenge,” Center for Strategic and Budgetary Assessments, Washington, DC, 2005.

[2] *U.S. Department of Defense Dictionary of Military and Associated Terms*, JP 1–02, Department of Defense, Washington, DC, 2010, p. 58.

[3] National Air and Space Intelligence Center, “Ballistic and cruise missile threat,” Wright-Patterson Air Force Base, OH, NASIC-1031-0985-13, 2013.

[4] D. M. Gormley, “Missile defense myopia: Lessons from the Iraq War,” *Survival*, vol. 45, no. 4, pp. 61–86, Winter 2003.

[5] G. Gya, Ed., “NATO missile defence: Political and budgetary implications,” Centre for Democratic Control of the Armed Forces, Brussels, Belgium, Mar. 2013.

[6] Jane’s HIS. (2015, Apr.). Patriot PAC-3 record. [Online]. Available: https://janes.ihs.com.libproxy.nps.edu/CustomPages/Janes/DisplayPage.aspx?ShowProductLink=true&ShowProductLink=true&DocType=Reference&ItemId=+++1494652&Pubabbrev=JAAD

[7] Special feature. (n.d.). Ministry of Defense [Japan]. [Online]. Available: http://www.mod.go.jp/e/jdf/no11/special.html. Accessed May 2015.

[8] Jane’s HIS. (2004, May). PAC-3 ripple-fire test destroys ballistic and cruise-missile targets record. [Online]. Available: https://janes.ihs.com.libproxy.nps.edu/CustomPages/Janes/DisplayPage.aspx?DocType=News&ItemId=+++1198889&Pubabbrev=JMR

[9] Jane’s HIS. (2007, July). PAC-3 missile intercepts low-flying, air-breathing target in tests record. [Online]. Available: https://janes.ihs.com.libproxy.nps.edu/CustomPages/Janes/DisplayPage.aspx?DocType=News&ItemId=+++1197244&Pubabbrev=JMR

[10] Jane’s HIS. (2012, May). Integrated JLENS and PAC-3 shoot down surrogate cruise missile record. [Online]. Available: https://janes.ihs.com.libproxy.nps.edu/CustomPages/Janes/DisplayPage.aspx?DocType=News&ItemId=+++1506404&Pubabbrev=IDR

[11] R. Yanushevsky, *Modern Missile Guidance*. Boca Raton, FL: CRC Press, 2008.

[12] R. Broadston, “A method of increasing the kinematic boundary of air-to-air missiles using an optimal control approach,” M.S. thesis, Dept. Elect. and Comput. Eng., Naval Postgraduate School, Monterey, CA, 2000.

[13] D. Pehr, “A study into advanced guidance laws using computational methods,” M.S. thesis, Dept. Elect. and Comput. Eng., Naval Postgraduate School, Monterey, CA, 2011.

[14] A.M. Osborn, “A study into the effects of Kalman filtered noise in advanced guidance laws of missile navigation,” M.S. thesis, Dept. Elect. and Comput. Eng., Naval Postgraduate School, Monterey, CA, 2014.

[15] P. Zarchan, Ed., *Tactical and Strategic Missile Guidance*, 6th ed. Reston, VA: Amer. Inst. of Aeronautics and Astronautics, 2012.

[16] R. Yanushevsky, *Guidance of Unmanned Aerial Vehicles*. Boca Raton, FL: CRC Press, 2011.

[17] S. N. Balakrishnan *et al.*, *Advances in Missile Guidance, Control, and Estimation.* Boca Raton, FL: CRC Press, 2013.

[18] U. S. Shukla and P. R. Mahapatra, “The proportional navigation dilemma—Pure or true?,” *IEEE Trans. on Aerospace and Electron. Syst*., vol. 26, no. 2, pp. 382–392, March 1990.

[19] R. G. Hutchins, “Navigation, missile, and avionics systems,” class notes for EC 4330, Dept. of Elect. and Comput. Eng., Naval Postgraduate School, Monterey, CA, January 2005.

[20] Y. Bar-Shalom *et al.*, *Estimation with Applications to Tracking and Navigation: Theory Algorithms and Software*. New York, NY: John Wiley & Sons, 2001.

[21] R. G. Hutchins, “Optimal estimation, Kalman filters and target tracking,” class notes for EC 3310, Dept. of Elect. and Comput. Eng., Naval Postgraduate School, Monterey, CA, March 1997.

[22] A. Bradley and C. Duffy, “Missile interceptor,” Georgia Inst. Technol., School of Aerospace Eng., Dec. 12, 2011, p. 29.

[23] G. P. Sutton and O. Biblarz, *Rocket Propulsion Elements*, 8th ed. Hoboken, NJ: John Wiley & Sons, 2010.

[24] Federal Aviation Administration, *The Pilot’s Handbook of Aeronautical Knowledge*, U.S. Federal Aviation Administration Flight Standards Service, FAA-H-8083-25A, Oklahoma City, OK, 2008, ch. 4.

[25] J. J. Gottlieb, “External pulse effects on solid rocket internal ballistics,” AIAA, Huntsville, AL, 2000

[26] Naval Research Laboratory, “Tomahawk Cruise Missile Flight Environmental Measurement Program,” *The Shock and Vibration Bulletin*, Washington, DC, 1982.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California